THE UNIVERSITY *of* EDINBURGH

Understanding Requirements Work in E-Science Projects

Ka Lai Ho

Doctor of Philosophy

Institute for Communicating and Collaborative Systems

School of Informatics

University of Edinburgh

2013

Abstract

The e-science vision is to create infrastructures to enable faster, better and more collaborative science to be carried out in the 21st Century. The goal is for these infrastructures to allow scientists to collaborate routinely, scaling geographical and disciplinary boundaries; to create ad hoc arrangements datasets, equipment or computational power to solve larger, more complex scientific problems; to federate remote datasets, hence, aiding scientists in data discovery and even data re-use. The work to turn the e-science vision into reality has been the subject of major research programmes in the UK (UK E-Science Programme) and the US (National Science Foundation's Cyberinfrastructures Programme). Inevitably, e-science technologies and scientific practices will co-evolve as collaborative work becomes more prevalent, and cross-disciplinary work becomes routinised. Thus, the design and development of e-science technologies will play a critical part in the above process; there is a clear need to develop technologies which will accurately reflect end-users' needs as well as account for the wider social structures of future scientific work.

In e-science, there has been an on-going debate about whether new requirements techniques are needed to deal with the 'unique' characteristics of e-science projects, and the ambitious aims of e-science software. Some argue that e-science presents sufficiently novel combinations of challenges that new techniques are needed, whilst others argue that e-science practitioners should 'borrow' from pre-existing requirements engineering techniques. However, one barrier in settling this question (insomuch as it can be 'settled') is that there is currently a lack of empirical data on the requirements work and activities carried out in existing e-science projects. Studying requirements activities in e-science projects by examining the actual problems encountered would yield insights regarding the challenges in working out requirements for e-science technologies, and more generally, better inform the structure of requirements work in future projects.

The research in this thesis examines the requirements activities of three UK e-science projects drawn from the astronomy, molecular simulation and translational science disciplines. Detailing the experience of project team members, the research explores issues and challenges encountered over the course of working out requirements for e-science technologies. In particular, this research takes a slightly different approach from similar studies, with the unit of analysis being at the project level, reflecting shifts in emphasis on requirements work as projects evolve.

Three aspects of requirements work in e-science projects are explored closely. Firstly, the temporal patterns in development work and how requirements activities fitted into such rhythms, phases and trajectories; secondly, the challenges of making a prototyping approach work; thirdly, the challenges of stabilising the 'missing middle' – a term to describe the gap between the high-level visionary description of the system from the project proposal, into fine-grained, detailed requirements. Then all three themes are drawn together, in order to make more general observations regarding the challenges of working out requirements for e-science technologies, as well as some observations regarding the shape of requirements work over the course of an e-science project.

The thesis concludes that working out requirements for e-science technologies is challenging due to the complexities of supporting multi-disciplinary and multi-organisational work and the novelty of the technology to be developed. Team members have to grapple with multiple domains of knowledge, where there is little pre-existing expertise on how these aspects could be combined. Evidence from the data suggests that multiple strategies are employed to manage this complexity; where the selection of techniques is done on a contingent basis. Thus, one of the major implications of this thesis is to suggest more systematic and explicit capture of 'lessons learnt' from developers of previous e-science technologies.

Acknowledgements

First of all, I would like to thank my supervisors Stuart Anderson and Mark Hartswood, who have been incredibly patient and encouraging throughout my PhD. I have learnt a tremendous amount from both of them – on life, the universe and everything. Needless to say, this thesis would not have been possible without them.

I also want to take this opportunity to thank Rob Procter, who shaped much of the start of this research. He helped to locate funding for me to undertake this work and I am very grateful for his input.

Of course, I also want to thank my colleagues at the Social Informatics Cluster throughout the years: Jenny Ure, Alex Voss, Roger Slack, Massimo Felici and Conrad Hughes. I want to thank Jenny for being always happy to help (and her dedication to work never ceases to amaze me); Alex and Roger (along with Mark) for giving me an appreciation of the British sense of humour; Conrad and Massimo for musings over software development, as well as numerous other topics that I won't list here.

Then there are the people who gave up their time to proofread this (or numerous other versions of this) thesis – Ianthe Hind, Mark Sutherland, Clare Huxley, Tim Willis and Jo Wells. I want to thank you all for all of your help and encouragement.

Finally, I want to give special thanks to various friends and family, who, have supported me in a in a variety of ways throughout this thesis. I am extremely grateful to be blessed with such understanding, caring and inspirational people to be around. Thank you for being there.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Ka Lai Ho

Contents

Abbreviations

AG – AstroGrid

EPSRC - Engineering and Physical Sciences Research Council

GridPP – UK Particle Physics Grid project

PI – Principal Investigator(s)

NERC – Natural Environment Research Council

NVO – National Virtual Observatory (U.S. project)

PPARC – Particle Physics and Astronomy Research Council (now Scientific Technology and Facilities Council)

SOA – Service Orientated Architecture

SOPs – Standard Operating Procedures

VO – Virtual Organisations

VOb – Virtual Observatory

Chapter 1: Introduction

1.1. Background

The desire to build novel, large-scale software infrastructures to support multi-disciplinary and multi-organisational collaborative work has recently become popular. Large-scale programmes such as Connecting for Health[1] (an ambitious, multimillion pound project by the National Health Service to create an integrated care record system) promise 'all singing, all dancing' systems which will enable integrated access for data from multiple sources, and allow all practitioners to conduct collaborative work on a routine basis. This trend has been driven by the prevalence of the internet and network technologies in everyday life; increasing the expectations of what is possible in terms of connectivity and enabling collaboration.

Similar ambitions for enabling technology have been seen in science. The increasing use of networking technologies has seen gradual changes in scientists' domestic and work lives. Scientists can use newsgroups and mailing lists to keep in touch with the latest news in their fields, outwith large conference meetings. They now have on-

[1] http://www.connectingforhealth.nhs.uk/

demand access to scientific papers through online publisher portals rather than needing to obtain paper copies, and they can transfer their results to share, and discuss through email, with colleagues halfway around the world. All this, has increased the expectations of changes that communication and collaborative tools could enable in the scientific community (Walsh & Bayma, 1996). Spotting this trend, a number of advocates championed the opportunity to be able to carry out a more systematic approach to enabling multidisciplinary, multi-organisational, collaborative work in science. This endeavour turned into the UK e-science programme (Hey & Trefethen, 2002; Foster, 2002), and the U.S. cyberinfrastructure programme (Atkins, 2003).

The e-science vision promises to enable faster, better and more collaborative science for the 21st Century (Hey & Trefethen, 2002). The goal is to build computing infrastructures that will enable scientists to collaborate routinely, scaling geographical and disciplinary boundaries, to create ad hoc arrangements datasets, equipment or computational power to solve scientific problems and to federate remote datasets, hence aiding scientists in data discovery and even data re-use. If realised, the e-science vision would enable scientists to tackle larger, more complex scientific problems in a multi-disciplinary manner. It is this promise of enabling faster scientific advances that makes e-science such an attractive proposition. As a result, the UK e-science programme was created to systematically create the technologies and the computing infrastructures to realise the e-science vision (Hey & Trefethen, 2002). As part of the e-science programme, Grid computing, a computing paradigm that manages heterogeneous resources in an integrated manner was hailed as its enabling technology (Foster, 2002; Foster, Kesselman, & Tuecke, 2001). Thus, a significant proportion of the UK e-science programme was to deliver a national Grid infrastructure and middleware tools.

Whilst one aspect of the e-science programme has been to put in place the large-scale, national computing infrastructures, another integral component was the development of discipline-specific software tools to enable the everyday scientist to

work in an "e-science manner". The aim was that these software tools would be tailored to the individual discipline's primary modes of scientific enquiry (e.g., astronomers have different needs to computational chemists). This tailoring allowed the gap to be bridged between the generic Grid infrastructure and the scientific disciplines, hence providing the metaphorical science "pull" to complement the technology "push" (Hey & Trefethen, 2002). Thus, significant funding was provided to each of the UK research councils to create e-science tools ranging from proof of concept prototypes to large-scale production level software infrastructures, to enable scientists to conduct the multi-disciplinary, multi-organisational collaborative work envisaged by e-science. As one can imagine, the success of the e-science programme is heavily dependent on the sustainability and adoption of discipline-specific e-science technologies. This is because it is only by providing the tools to enable the scientists' work that the e-science programme will successfully bring about changes to the working practices of individual scientists.

For a variety of reasons developing e-science applications and infrastructures is challenging. Firstly, the software has to accurately reflect end users' needs as well as to account for the wider social structures of future scientific work. Secondly, the software has to be built for a heterogeneous population of users using a variety of tools, and in a variety of configurations. E-science applications are expected to be implemented with Grid middleware, a novel and evolving computing paradigm. On top of this, e-science projects tend to have a distributed development model, and distributed software development is only just beginning to be studied in software engineering literature. It is this combination of the above characteristics that makes understanding the requirements of these technologies such a difficult endeavour. Coupled with all these factors, is the wide acknowledgement that technological advances alone are not enough; in order to realise e-science to its full potential, significant changes to the organisational and social structure of scientific culture will have to occur. Welsh, Jirotka and Gavaghan (2006) for instance, point to the individualistic reward structures in science as a barrier to encouraging collaborative work between scientists. In this context, understanding the requirements for e-

science technologies is not simply about "capturing" the requirements for a well-defined or well-known problem. Instead, developers will have to account for an evolving set of working practices and a changing environment, where the software tools and the work practices will co-evolve over time.

Because of the sorts of challenges and complexities outlined above, some researchers have argued that traditional requirements and usability techniques are inadequate in dealing with the demands of implementing e-science technologies (Zimmerman & Nardi, 2006), while others have called for the e-science community to 'borrow' from other disciplines (e.g., requirements engineering, software engineering and workplace studies (Voss, Procter, Jirotka, & Rodden, 2007)). Whilst it remains unclear whether new requirements techniques are needed for e-science, one barrier to getting a better grasp of the issues and challenges associated with the requirements work involved is the lack of empirical data upon the subject. Considering the number of researchers that have theorised about how challenging requirements and software development work is in the context of e-science, there is a surprising lack of empirical data regarding how requirements work is currently being carried out (recent exceptions being de la Flor et al. (2007)).

Understanding requirements work is critical to the development of e-science technologies, since grasping 'what to build' is the first step in any software development. If requirements for e-science technologies are so difficult to carry out, it is reasonable to be concerned about the following question: *Is the e-science community failing to produce software tools that accurately reflect the needs of the scientists?*

This is an important question in the long term; as long as there is a trend towards collaborative science, there will also be a need to find effective requirements techniques for such technologies.

This thesis aims to contribute to existing literature by providing a strategic overview of requirements engineering challenges in e-science technologies and projects. To do

this, a qualitative approach was taken, with three case studies being explored drawn from astronomy, computational chemistry and translational cancer research. Data collection was through a combination of semi-structured interviews and observation. In this study, the unit of analysis was at the project level.

One aim of this research is to move on from simply theorising about the challenges associated with e-science projects, to a discussion of the realities of software development work. This includes provision of empirical "on the ground" accounts of the issues that arise over the course of requirements work, with the focus on detailing how such challenges are dealt with in practical terms. Consequently, this research can be used to a) suggest the structure of future requirements activities for e-science projects, and on a more practical level, b) provide a resource for future e-science participants to draw upon in anticipating the sorts of problems that they might encounter.

As such, the research questions being investigated in this thesis are:

- *How are requirements activities currently organised in existing e-science projects?*
- *Do existing e-science project teams draw upon established requirements techniques and if so, when?*
- *What characterises the difficulties in articulating requirements for e-science technologies?*

This thesis aims to show that developing software for e-science is not an established, routine activity; instead, the team members in the projects studied had little prior expertise available to draw upon. Hence they had to learn *how* to build e-science infrastructures, components or applications *whilst* building them. Thus, this thesis argues for a more sophisticated view of requirements (and its articulation) for e-science technologies and projects,

1.2. Overview of Thesis

This thesis is divided into three main parts. Firstly, Chapters 2 and 3 provide background and a review of the literature, with Chapter 4 bringing into focus the research methodology used. Secondly, Chapters 5, 6, and 7 are the data chapters of the thesis. Finally, Chapter 8 reviews and summarises the findings, and discusses implications and future work.

Chapter 2 provides the background to e-science. The chapter starts by outlining the e-science vision, and describes the drivers that prompted advocates of e-science to create the first systematic attempt at instantiating e-science: the UK e-science programme. The focus then shifts onto detailing the e-science programme, from which two out of the three case studies are drawn. It gives a sense of the scale of the effort needed to realise e-science, and also describes the wider context of the e-science programme within which the case studies are situated. Whilst the e-science programme promises a paradigm shift in scientific work, turning the vision into reality will not be an easy task. There are significant technical challenges to overcome, and shifts in social and cultural structures in science will be necessary, before it is possible to enable routine multi-disciplinary, multi-organisational collaborative work. On the technical side, one of the major hurdles is the creation of the technical architecture to facilitate seamless sharing of data and resources across multiple organisations. Grid computing and later, web services were hailed as the enabling technologies for e-science because their architecture fitted this goal. A brief outline of Grid computing and web services are provided in this chapter.

Chapter 3 presents the literature behind the main problem addressed in this thesis, i.e., understanding the challenges of articulating requirements for e-science technologies. In the e-science and cyberinfrastructure communities there has been an ongoing debate regarding whether e-science technologies present a sufficiently ‘unique’ challenge, when compared with previous software development projects, to warrant the creation of new requirements techniques, or whether pre-existing

techniques are adequate for this purpose. Consequently, the chapter begins with a discussion of the combination of challenges that characterise e-science projects and software in existing literature. This thesis adds to the small but growing body of work discussing requirements techniques for e-science, by examining three e-science projects drawn from the disciplines of astronomy, computational chemistry and translational cancer research. The issues and challenges associated with understanding the requirements for e-science technologies are explored from the perspective of the project team, alongside accounts of how requirements activities are situated in e-science projects.

Chapter 4 presents the research methodology behind the thesis. It describes the use of the case studies, the criteria behind each of the projects studied, the data collection methods used, and the data analysis process.

Chapter 5 is the first of the data chapters in this thesis and describes the AstroGrid case study. AstroGrid was the UK's implementation of the Virtual Observatory (VOb), and was one of the biggest projects in the e-science programme. The astronomy community has a history of building large-scale hardware scientific instruments in a distributed manner. AstroGrid, according to one of the project members, was one of the largest (if not the largest) single collective software projects undertaken by the UK astronomy community. The astronomers approached building the VOb using their expertise from previous hardware projects. The plan was that the purpose for the instrument would be outlined, from this a detailed requirements specification would be drawn up, and design and implementation would follow. As the project progressed, however, it became clear that building large-scale software infrastructures required a very different approach, so they hired software developers with industrial experience and had to improvise a software development process that met their needs processes to build the VOb. The data revealed different sorts of temporal patterns in the development work: rhythms, phases and trajectories, and how requirements activities are situated in each of these patterns. It was found that certain types of requirements were emphasised as the projects progressed and that

developers have to account for the wider landscape when considering requirements for their infrastructures.

Chapter 6 describes the findings from the eMinerals case study. A common recommendation in e-science design literature is to use prototypes and prototyping approaches to understand the scientists' needs; however, little guidance is offered as to how a prototyping approach might be carried out in practice. The eMinerals project provides insight into the difficulties encountered in making a prototype approach work in practice. eMinerals presented an altogether different type of project compared with AstroGrid. Its aim was to provide a testbed to evaluate the suitability of Grid middleware for the types of scientific research that molecular simulation scientists were conducting. The majority of their requirements elicitation was carried out using prototypes in an iterative development process, where the project scientists were asked to test out a tool and provide feedback on whether it was useful to them or not. Despite being a computation-based discipline, where most of the scientific work was conducted using high performance computers, simulation code and scripting interfaces, the transition to Grid technologies and the e-science paradigm was not altogether an easy one. Instead, a significant amount of work, including a careful choice of problem, was needed at the start of the project in its planning and structure, as well as throughout the project, in order to make the prototyping approach a successful one.

Chapter 7 presents the final case study considered in this research, and provides more detailed, day-to-day accounts of requirements work in e-science. The chapter explores the complexities involved in 'working out' requirements for e-science, and detailing the strategies by which requirements get 'stabilised' as the project progresses. The data is drawn from a 10-month observation-based study. TransProject was a data infrastructure project with the aim of building a dataset to conduct epidemiological research on patients diagnosed with colorectal and breast cancer within a city hospital. The project was the smallest of the three outlined here, and unlike the other two case studies, the fieldwork period spanned the initial year of

the project, charting the trials and tribulations of understanding the 'problem' and of arriving at feasible 'solutions' associated with working out requirements for their system. The idea here is that the transition between the project vision and the detailed requirements is challenging because the project team members had to account for multiple, diverse domains of knowledge in order to work out feasible solutions, and also had to engage different requirements elicitation strategies in order to try to solve particular problems. Often it was not clear what a desirable, or even feasible, solution would be due to the complexities of working across multiple knowledge domains and multiple organisations. The latter part of this chapter takes the idea further, and identifies five strategies that team members used to attempt to stabilise the requirements. These strategies are described and illustrated through empirical data from the project.

Finally, Chapter 8 contains the discussion and conclusion. The discussion in this chapter draws on the findings from the case studies, and considers them from the perspective of understanding how requirements work is conducted in the design and development of e-science technologies. The findings suggest a blurring of the boundaries between what is traditionally characterised as "requirements" work and that characterised as "design" work. In the context of e-science, therefore, it was found that the participants often oscillate between defining what the requirements are for the tool and building a small prototype, before using it to redefine the requirements. This chapter also presents the implications for e-science practitioners, funders, and, requirements engineers followed by implications for further research, and a section reflecting on the research methodology.

Chapter 2: E-Science – From Vision to Reality

2.1. Introduction

E-science is a grand vision that promises to create computing infrastructures and applications to enable faster, better and more collaborative science to be undertaken in the 21st Century (Hey & Trefethen, The UK e-Science Core Programme and the Grid, 2002). If realised, the benefits of e-science would include the ability to tackle larger, more complex scientific problems on a routine basis.

The goal of this chapter is to describe the e-science vision and provide background to the motivation behind designing and developing e-science technologies. The purpose is to provide context to the arguments that will be made in the subsequent chapters; namely that requirements for e-science technologies are emergent from the dialogue between developers and scientists. To do so, the chapter starts by outlining the wider context in which the research lies: starting with a brief description of the e-science vision and the growing trends that which have made e-science such an attractive proposition. It is important to note that throughout this and the subsequent chapter the underlying assumption is that the adoption of e-science technologies will play an integral part in the success of the e-science programme's goals. E-science technologies – a term commonly used in this thesis to represent the software enabling the scientists to work in an 'e-science manner' – if adopted widely outside of the e-

science programme, will likely mean that the philosophies and goals behind e-science will be sustained long after the initial funding programme. In order to ensure that these tools are adopted, and more importantly, 'embedded' (de la Flor et al., 2007) into the scientists' work, there is a need to ensure that these e-science tools accurately reflect the needs and desires of the scientists using them. Thus, understanding the requirements engineering process behind these tools is an important part of ensuring the sustainability of the e-science paradigm as a whole.

2.2. E-Science: The Rise of Large-Scale Collaborative Science

> *"Each of the [e-science] projects had great aspirations for bringing about serious change in the world of science and engineering by facilitating new approaches to data analysis, the extraction of knowledge from very disparate sources, and the application to experiments that were until now unimaginable.* ***The future of large-scale, collaborative science is here now*** *…" (EPSRC Report on E-Science, 2004; emphasis added)*

E-science is a grand vision which promises a radical transformation of scientific practices by enabling collaborative work across multiple organisations and different disciplines in an integrated manner, on a routine basis (Hey & Trefethen, 2002). Core to e-science, is the development of computing infrastructures that facilitate multi-organisational sharing and seamless access to resources. The claim of the proponents of e-science is that these infrastructures will enable the rapid assembly of resources in an ad hoc manner to solve scientific problems, where resources could include data, computational power and equipment. These infrastructures will also facilitate the linkage and federation of large disparate datasets, providing scientists seamless access to data. Plus, it will allow rich communication and collaborative tools to run that will allow scientists' work to scale across geographical distances. During the early e-science programme, Grid computing was hailed as the enabling technology of e-science; a further explanation of Grid technologies is detailed later on in this

chapter (section 2.5.1). Advocates claim that e-science applications and infrastructures will enable scientists to do bigger, faster and better science.

In order to understand the motivations for e-science, one has to look at the wider trends in scientific work. The vision of e-science is deeply rooted in the increasing use and prevalence of IT and networking technologies in general. In the scientific community specifically, the use of the network technologies has become increasingly prevalent. Technologies such as the internet have been slowly adopted and gradual shifts have been seen as these communication technologies have become a more prevalent part of domestic and work life for scientists (Walsh & Bayma, 1996). This trend has, in turn, raised expectations in the scientific world regarding interconnectivity, and the ability to access remote resources on a routine basis. With internet and email, scientists have online access to journals and other papers, whereas previously they would have had to access a paper version supplied by the author, or the university library. Scientists can now communicate in real time with colleagues halfway across the world, without physically relocating or incurring significant charges, which would have been impossible before the advent of the internet. In addition, the use of bulletin boards has enabled academic discussions and debates to take place virtually, supplementing discussions at conferences (Walsh & Bayma, 1996).

Advocates of e-science, not only saw this underlying drive towards the use of more communication technologies, but they also argued that there were several other trends that were occurring in scientific research which motivated the development of e-science infrastructures. These trends included the need to manage increasing volumes of data, the increasing demand for computational power, and lastly, the rise of collaborative science.

Firstly, there are claims that increasing capacities of storage devices and networks have allowed the creation of large data archives that act as the driver behind data intensive science. Such large data archives have resulted from advances in equipment enabling more data to be captured at once, and also from more complex simulations

allowing more data to be generated. Having larger data archives means there is possibly a need to produce effective data management tools to discover, analyse and visualise that volume of data (Berman & Hey, The Scientific Imperative, 2004).

In addition, it is conjectured that the decreasing cost of data storage has allowed the retention of data that might previously have been thrown away. Thus, large data archives can be built up with the potential for future reuse for different purposes. One claim is that these data archives can quicken the research process, as old data can be reused in subsequent scientific studies. Thus, there is great interest in building large-scale data platforms to a) federate disparate datasets (i.e., link them), and b) facilitate the storage and re-use of data.

Secondly, it is argued that there is an increasing demand for computational power in scientific work, partly driven by the rising popularity of computational simulations as a mode of scientific enquiry (Berman & Hey, The Scientific Imperative, 2004), and partly as a result of the need for computational power for data analysis of large datasets[2]. According to Hey and Trefethen (2003, p. 810) "in many planned and future experiments they [the scientists] are also planning to generate several orders of magnitude more data than has been collected in the whole of human history". Whilst previously, the ability to manage experimental data with significant trends/findings was (generally) manageable; the dramatic increase in the size of datasets will render this near impossible without automation. Thus, the need for computational power to analyse large datasets is an important driver for the adoption of e-science.

---

[2] In particular, having access to large data archives will mean little without advanced data analysis techniques to accompany them. Manual analysis was possible with small volumes of data; however, predicted volumes of data make manual analysis (near) impossible. Advances in information processing techniques will require and drive the demand for computational power. For example, GridPP is building the infrastructure as part of the UK's contribution to the LHC, and will need to handle the storage and data mining resources of petabytes of data (equalling 1000 terabytes) per year (Hey & Trefethen, 2003). In order to analyse the large volume of data produced, significant computational capabilities are required (ibid).

Lastly, the increased availability and capacity of high-speed networks has meant that it is possible to consider the use of remote resources rather than simply "local" resources (Foster I. , 2002). Having remote access to data, equipment, or computational power means that it is possible to consider coordinating local and remote resources, rather than having to transfer all of the data onto the local machine to conduct data analysis. In e-science, collaboration arrangements of this sort are referred to as "virtual organisations" (Foster, Kesselman, & Tuecke, 2001).

Some members of the scientific world, especially those who have had experience in the large-scale, 'Big Science' disciplines such as Particle Physics (Galison & Hevly, 1992), have seen the combination of these trends as an opportunity to take a more systematic approach to developing a computing infrastructure to support future scientific research (Freeman, 2007). John Taylor, the Director General of Research Councils in 2000, saw the potential for large-scale, multi-disciplinary and distributed science to tackle larger, more complex scientific problems than could previously have been handled under a single discipline or institution[3]. He was also influenced by the concept of utility computing during his time working at Hewlett-Packard. In the same way that electricity or gas are utilities, and can be accessed on-demand, with utility computing, IT infrastructure would be universally available (for the individual/institution), and purchased as a service rather than having to purchase the infrastructure outright[4] (Hey & Trefethen, 2002). Coupling these two visions, Taylor successfully bid for a programme to create a large-scale infrastructure programme intended to support collaborative, multi-disciplinary science, which became known as the UK e-Science programme.

---

[3] This can be potentially attributed to a lack of resources that can be made available through one institution, or there might not be enough expertise or knowledge to solve the problem within one discipline.

[4] It is possible to argue that this idea is not new – especially in highly coordinated disciplines such as High Energy Physics (HEP) where there are suggestions that the infrastructure is not owned by the scientific institutions themselves, but rather the institutions purchase the use of the infrastructure.

2.3. UK E-Science Programme

2.3.1. Introduction

> *"e-science is about global collaboration in key areas of science and the next generation of infrastructure that will enable it." John Taylor, the Director General of Research Councils at the Office of Science and Technology*

The UK's e-Science programme was the first of its kind in creating a systematic agenda to implement the infrastructure to enable e-science to occur. Funded from 2001-2006, and costing £194 million, the UK programme is one of the largest e-science programmes to date (Hey & Trefethen, 2002), and sparked off a series of similar programmes globally (see 2.3.4. Other Programmes). This section outlines the details of the UK programme, with particular emphasis on the pilot projects. This is because two of the case studies in this thesis are drawn from this programme. The aim of this section is to provide the reader with a sense of the scale of the effort in the e-science programme, and to provide a flavour of the wider landscape of e-science and Grid technology developments within which the case studies in this thesis were situated. This section draws heavily on Hey and Trefethen (2002), which outlines the programme.

2.3.2. First Phase

In November 2000, the UK government announced funding of £98 ($180) million for the UK E-Science Programme to be managed by the Engineering and Physical Sciences Research Council (EPSRC). Broadly speaking, the programme consisted of two parts: a set of *pilot projects* managed through each Research Council, and an infrastructure and support base through the *Core Programme*. These are identified in

Figure 1 as the Academic Application Support Programme, the Generic Challenges, and Industrial Collaboration respectively.

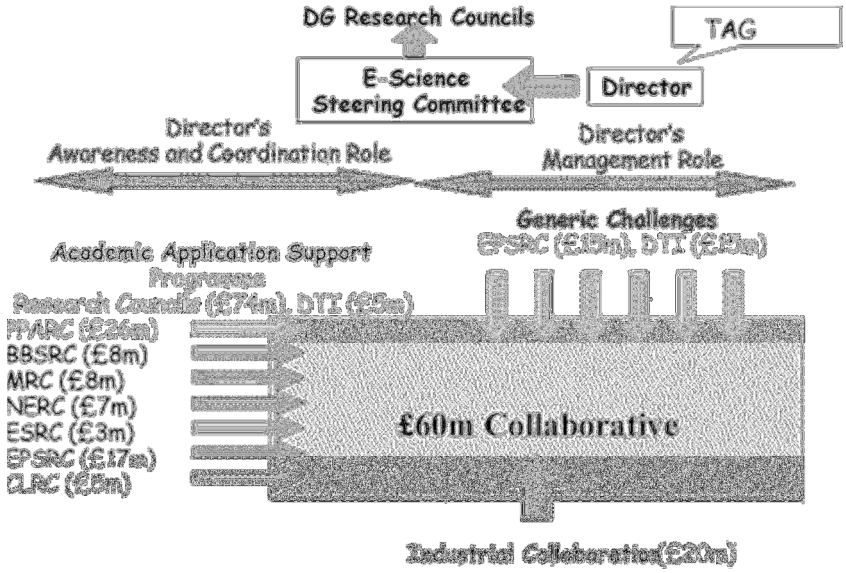


Figure 1: Funding for UK E-Science Programme (Hey & Trefethen, 2002)

The programme was effectively split into two halves, both complementary to each other. The first half was the development of real tools and applications that scientists could use. This was achieved through a series of Research Council led pilot projects. While these tools would make a difference to the scientists on the surface, it was recognised that these tools required access to powerful computation and data infrastructures. Thus, the second half of development became the E-Science Core Programme, which was responsible for building a national infrastructure that could power e-science tools from all disciplines.

Collectively, they represented a multi-faceted approach to realising e-science. The pilot projects provided science-based applications for specific scientific disciplines and would act as 'real world' demonstrations of what e-science could achieve. The Core Programme took generic requirements and technical needs derived from the pilots, and alongside industrial involvement, aimed to produce robust middleware for both research and industry. Lastly, the programme had a clear remit to engage with

industry, in knowledge transfer and to build tools/expertise in e-science and Grid technologies. It was claimed that this structure would enable the e-science programme to gain more rapid results than traditional funding programmes.

The UK E-Science programme was funded in 2 phases. The first phase of the programme was funded from 2001 – 2004. At the time, it was the first programme of its kind to focus solely on e-science, and arguably, subsequently inspired a number of e-science-like programmes around the world. Participants in the programme saw it as unique. It acted as a funding programme that had the sole purpose of developing scientific tools for research; something that had been difficult to do under traditional research funding. As seen later on in the data chapters, most scientists that worked within the e-science programme saw it as a separate funding stream[5]; obtaining funding to build scientific tools was difficult to justify as "research" to the Research Councils. This would be even harder should those tools require expertise spanning beyond the remit of a single Research Council.

Each Research Council was allocated funding for Pilot Projects. Councils were free to choose how funds were distributed, and provided their own selection criteria, their own allocation process, and they alone also decided how many projects were to be funded. The number of projects chosen varied amongst the Research Councils, for instance, PPARC only supported two large projects – GridPP (Particle Physics) and AstroGrid (Astronomy), whilst EPSRC's funding was distributed amongst six pilots: RealityGrid, Comb-e-Chem, DAME, GEODISE, DiscoveryNet, myGrid. Some Councils chose projects in terms of subject of the proposals; for instance, EPSRC did not specify particular areas, whilst BBSRC's projects built upon their current strengths in "post genomics, structural biology/structural genomics" (Hey and

[5] This was also a point made by Schroeder and Fry (2007)

Trefethen, 2002). Other criteria included the organisational structure of the projects and outcomes of such projects; EPSRC, for example, specified that projects had to be of a consortium nature and able to produce working grid systems at the end of the three year funding period.

From these pilot projects, it was anticipated that requirements for a cross-disciplinary, national Grid infrastructure could be gathered (details of Grid infrastructure to be discussed later in 2.4). These requirements would facilitate middleware development and would cut across all disciplines, such as data curation, security etc.

Pilot Projects varied in size, scope and ambition dependent on their discipline. For instance, GridPP and BioSimGrid, were two funded pilot projects. These are outlined in Table 1.

In addition to the Pilot Projects, the e-Science Core Programme was also integral to the e-Science programme. A total of £15 million was allocated to the e-Science Core Programme, which EPSRC was responsible for (RCUK Website, 2008). As outlined in Hey and Trefethen's (2002) paper, it had a dual role of support and development. On the support side, the e-Science Core Programme was chiefly involved in providing technical support to each of the pilot projects, and acted as a knowledge hub for Grid based enquiries, such as setting up Grids. This support was not limited to the Pilot Projects, but was also available to the industrial collaborators. With respect to development, generic requirements from the Pilot Projects were gathered (generic in the sense that they were non-discipline specific), and provided the basis for development projects within the programme. The emphasis was placed upon the developmental side, with the subtext that it would provide a good basis for industrial collaborations:

> *"The goal of the Core Programme is to support the e-science pilot projects of the different Research Councils and work with industry in developing robust, 'industrial strength' generic grid middleware. Requirements and lessons learnt in the different e-science applications will inform the development of more stable and function grid*

middleware that can assist the e-science experiments and be of relevance to industry and commerce." (Hey and Trefethen, 2002)

In some ways, the Core Programme served as the 'glue' between all the Pilot Projects, since it helped with the knowledge transfer of generic problems from one discipline to another. This point is actually quite important later on (Chapter 6), since the solutions engineered for one particular purpose (e.g., metadata management) were seen to be applicable for other disciplines. In practice, it facilitated a community built around e-science.

Table 1: Two examples of e-science projects

BioSimGrid (BBSRC)
The BioSimGrid project is run by six different university labs across the UK, and its aim is to utilise Grid infrastructure in order to share the results of large scale biological simulations. BioSimGrid is concerned with simulation data that "exists in the form of trajectories (sets of coordinates corresponding to the positions of atoms for a series of time steps) which can go up to the size of 10 GB per trajectory" (Ng et. al., 2006). The project has produced a prototype that has 20 users, and contains 200 trajectories with 1 Terabyte of files, with an additional 450Mb of metadata. The system allows users to submit data into the repository, resulting in it being made available to the entire community. An additional requirement was that the project would make data retrieval from the database usable and simple, so that users do not have to understand the underlying database infrastructure in order to conduct a query, or retrieve a record from the database (ibid). The project has the following requirements (ibid:658): (i) to provide data transparency, so that it is possible to show the location of the data, (ii) to ensure that the data transfer rate is maximised, (iii) to build an abstraction so that scientists do not have to learn database query languages in order to query data and (iv) to provide an analysis for the infrastructure.

GridPP (PPARC)

GridPP is a £33 million, six year project which is the focus of the UK's effort to create a grid infrastructure for the Particle Physics community. This is one of the e-Science flagship Pilot Projects and was built especially in preparation for the storage and analysis of the data produced by the Large Hadron Collider (LHC) at CERN. The overall aim of GridPP (from www.gridpp.ac.uk/vision.html) is "to develop, deploy and operate a very large Production Grid in the UK for use by the worldwide particle physics community."

The Core Programme was split into six activities (Hey and Trefethen, 2002):

1. Implementation of a National E-Science Grid testbed based on a network of e-Science centres
2. Promotion of Generic Grid middleware development
3. Interdisciplinary Research Collaboration (IRC) pilot projects
4. Establishment of a support structure for e-Science projects
5. Support for involvement in international activities
6. Support for e-Science networking requirements

In addition, part of the programme was collaborating and working with industry. The planned programme included heavy involvement from industry, and contained funding from the Department of Trade and Industry (DTI) that had to be matched by industrial partners. Around £20 million were raised by industrial collaboration. A few of the Pilot Projects reflected this link. For example, the DAME project (Distributed Aircraft Maintenance Environment) was a collaboration with Rolls

Royce, and Data Systems and Solutions. The project focused on "performance issues such as large-scale data management with real time demands" and in general, distributed diagnostics.

2.3.3. Second Phase

The second phase of the programme was funded in 2004, and continued until 2006. It received £96 million pounds of funding, with £16 million made available to the research infrastructure, and £11 million to the Department of Trade and Industry (DTI) technology fund (BBSRC, 2004). Each of the Research Councils was allocated additional funding for Pilot Projects, which resulted in continued funding for selected projects from the first phase, or in new Pilot Projects being set up. For example, two of the projects in the case studies had continued funding in the second phase of the projects, increasing each three year project to six years. New Pilot Projects set up during this phase were commonly known as the second generation of e-Science projects. It was largely acknowledged that they benefited from the first generation of projects either through the technology produced (for instance, the myGrid workflow engine – Taverna[6] – was used by the second phase Integrated Biology project (Mascord, 2006)), or through the expertise/experience gained (people from previous e-science projects working in the next generation projects).

The second phase of the e-Science core programme centred around six activities:

1. A national e-science centre linked to a network of regional e-science centres
2. Support activities for the UK e-science community[7]

---

[6] http://taverna.sourceforge.net/

[7] http://www.grid-support.ac.uk

3. An Open Middleware Infrastructure Institute (OMII)
4. A Digital Curation Centre (DCC)
5. New exemplars for e-science
6. Participation in international grid projects and activities

Of note, is the formation of the Open Middleware Infrastructure Institute (OMII), an organisation that acts as a central hub for knowledge for middleware (grid, web services or otherwise). It is based at the University of Southampton. Furthermore, the Digital Curation Centre (DCC) was created to facilitate research into digital data preservation and curation. It aims to provide software as well as advice to the e-science community[8].

2.3.4. Other Programmes

Using computational tools to facilitate scientific endeavours is not restricted to the UK. Similar initiatives have been taken elsewhere, with the biggest in the U.S. being based around the idea of *cyberinfrastructure* – namely the National Science Foundation's Cyberinfrastructures Programme. The focus in cyberinfrastructures is on computational infrastructures as networks to collate, mine and visualise large quantities of scientific data (see Ribes and Lee, 2010 for instance). While seemingly similar to e-science initiatives at first, those involved in cyberinfrastructure place much more importance on long-term aspects of these networks with emphasis on initial design, development and on-going maintenance of such structures. In turn, researchers have drawn attention to the tension between the characteristics of the computational world: namely, the fast moving nature of technological advances, and the characteristics of infrastructures: namely, considerations of reliability, upkeep and sustainability (Ribes and Finholt, 2009; Star and Bowker, 1999).

[8] http://www.dcc.ac.uk/about/

A list of e-science and Grid programmes (which had very similar aims) are listed in Table 2. There are two interesting points to note about the UK programme in comparison to the other e-science/Grid programmes listed. Firstly, the UK programme was the first programme of its kind, starting in 2001; and secondly, it was the biggest programme in terms of funding.

In summary, the UK e-Science programme had two strands – the Core Programme (generic Grid infrastructure) and the Pilot Projects (domain specific software tools and infrastructures).

Table 2: Table of Major Grid Initiatives – adapted from Gentzsch (2006)

Initiative	**Country of Origin**	**Time**	**Funding**	**People**	**Users**
UK e-science Phase 1	U.K.	2001-2004	$180M	900	Research
UK e-science Phase 2	U.K.	2004-2006	$220M	1100	Research and Industry
TeraGrid Phase 1	U.S.	2001-2004	$90M	500	Research
TeraGrid Phase 2	U.S.	2005-2010	$150M	850	Research
ChinaGrid Phase 1	China	2003-2006	$3M	400	Research
ChinaGrid Phase 2	China	2007-2010	$15M	1000	Research
National Research Grid Initiative 1	Japan	2003-2005	$25M	150	Research
National Research Grid Initiative 2	Japan	2006-2010	$40M	250	Research and Industry
Enabling Grids for E-Science 1	European Union	2004-2006	$40M	800	Research
Enabling Grids for E-Science 2	European Union	2006-2008	$45M	1000	Research and Industry
D-Grid 1	Germany	2005-2008	$32M		Research

D-Grid 2	Germany	2007-2009	$35M		Research and Industry

2.4. Technical Barriers: Enabling Technologies for E-Science

2.4.1. Searching for an Enabling Technology to Drive E-Science

As mentioned previously, a core part of the UK E-Science programme was the development of a set of networked technologies called Grid technologies. because the vision behind e-science technologies was to provide seamless access and work across a variety of existing computing infrastructure (Atkinson et al., 2003). At the start of the programme,, in computational terms, the complexity involved in coordinating data across organisations routinely is high. Unsurprisingly, this complexity arises from the heterogeneous combinations of software and hardware (which alone, are difficult to manage on a local level). To realise the e-science vision of seamless data sharing, tools have to be written allowing multiple configurations of networks to transfer data easily with each other. Additionally, this must be done in such a way that it allows users to log in from any of the computers in the network, and have a consistent and reliable user experience. A typical use case, would involve multiple scientists from the same project, logging in from different universities, and having all of the project's files display in a consistent manner. While this user experience might be possible to achieve within a single university, where the network configurations are generally known, this problem is acknowledged to be much more difficult once external universities are added.

Integration, in this sense, requires more than the transfer of data from one database to another, or from one computer to another. Instead, it extends much further, with the end goal considered at the end-user level rather than at the transaction level: data must be integrated across several databases and presented in such a format that it is meaningful for the scientist to analyse and use, from the perspective of answering scientific questions in the framework of the relevant discipline. As such, a 'simple'

action is likely to involve a series of complex instructions, calculations and computations that are hidden from the interface level. Atkinson, et al., (2003) outline different computational challenges that require computer science research in order to develop the "future infrastructure".

In order to drive the e-science vision then, a set of enabling technologies had to be found. These technologies had to act as 'glue' between networks.

Around the late 1990s, a new technology, termed Grid computing, showed great promise in potentially enabling the types of capabilities which e-science envisaged. Of particular interest, was the Grid's aim to manage dynamic large-scale, distributed and heterogeneous resources; meaning that Grid technologies could be built on top of existing organisations, and could link current resources together as and when they are needed. This management would be through the use of middleware – pieces of software that would orchestrate different resources together.

2.4.2. Introduction to Grid Technologies

The promise of Grid technologies was almost as grand as e-science. The term 'Grid' was used as an analogy to a power grid, but for computational power. From the user's point of view in the case of a power grid, electricity is available on-demand (from a wall socket), seamlessly (through a standardised plug) and can be accessed in a transparent manner (giving the impression of being a single power source). The consumer would simply buy and pay for electricity from the electricity companies. Grid technologies aim to do the same for computational power. Computational power and storage could be accessed in a distributed (regardless of geographical location), seamless (standardised, uniform) and transparent (single computer) manner:

> *"[w]hether the users access the Grid to use one resource (a single computer, data archive, etc.), or to use several resources in aggregate as a coordinated 'virtual computer', the Grid permits users to interface with the resources in a uniform way,*

> *providing a comprehensive and powerful platform for global computing and data management" (Berman, Fox, & Hey, 2003, p. 10)*

Therefore, consumers of Grid computational power, like consumers of electricity, would not have to build the infrastructure themselves. Instead, they would just be able to buy computational power like any other commodity or utility.

The premise was simply that instead of using high performance computers for computational power, middleware software could be installed on commodity Personal Computers, and be used to process data when they were idle. Individually, one PC would only provide a small amount of computational power, but collectively, with a few hundred, thousands or even tens of thousands of PCs a substantial volume could be provided. A piece of software would have to be installed on each computer, but once installed, the PCs could be managed from a range of places, as long as they were networked to the central node/management machine. Therefore, these machines can be spread across a number of different organisations; or in the case of a project such as SETI@home[9], people can 'donate' the idle cycles on their home computers for the analysis and processing of chunks of data allocated to them. In this context, Grid technologies can be distinguished from distributed computing, where the focus is on managing distributed resources; instead, the focus of Grid computing is on integrating multiple resources (Berman, Fox, & Hey, 2003).

As the Grid vision evolved, it expanded from simply accessing computational power, to the sharing of resources, including data, storage or instrumentation. Since Grid middleware managed heterogeneous resources, Grid could easily become more than simply for managing computational power; other resources such as dataset access and equipment can be managed using Grid technologies too. The promise of the Grid vision was that data could be federated and managed across multiple computers. The

[9] http://setiathome.berkeley.edu/

term, Virtual Organisations (VOs), is used to describe the assemblage of resources and individuals involved in "coordinated sharing" across multiple organisational boundaries (Foster, Kesselman and Tuecke, 2001). Grids could be of the data or computational type: depending on the main purpose intended for a particular Grid.

2.4.3. Challenges to Overcome in Building Grid Middleware

There are some significant challenges that need to be solved for the Grid middleware to function as desired. The middleware has to be able to coordinate and utilise the distributed resources in an efficient manner (Berman, Fox, & Hey, 2003). There are also unresolved issues such as quality of service, i.e., how to ensure that there is high performance across the network, scalability, i.e., how to manage potentially millions of individual resources, and interoperability, i.e., sharing protocols in a heterogeneous environment.

To demonstrate an instance of the challenges that exist, there are issues regarding the security policies for Grid Technologies (Atkinson, et al., 2002). For instance, *who would be allowed access to what? How can we enforce those policies?* Traditional models of authentication and security seem to be insufficient in a Grid environment, because existing security models work on the premise of client-server interactions, where both have to authenticate each other before data exchange can occur, whereas, in the Grid, nodes can act as servers or clients. It is suggested that new approaches to security are required – such as single sign-on, where a user only needs to authenticate once, even when utilising multiple resources. Alternatively, practical ways of defining authorisation to resources will have to be found, especially in a large community where it would be impossible in practical terms to define access rights for every single user.

Although Grid technologies have always been closely linked with e-science, they are not the only possible technologies to enable the type of integration of distributed resource management that e-science requires. Another set of technologies that was

emerging at around the same time (early 2000s) was Web Services and Service Orientated Architectures.

2.4.4. The Alternative Enabling Technology: Web Services and Service Orientated Architectures (SOAs)

A web service, in broad terms, is a resource that allows information or resources to be shared by machines (mostly automatically) in a distributed environment (Foster, 2002). The focus, in web service technologies, is upon automation. Web services are accessed and initiated, through automated means, between machines, rather than by humans. To look up what web services are available, lists of services are available in a 'directory' form, named registries. One of the main attractions of web services is that, as their name suggests, they can be transmitted over the web, making them universally accessible.

On a wider scale, web services can be used within a Service Orientated Architecture (SOA). An SOA is based around similar principles to Grid technologies, being a distributed system architecture based around services, where each service is owned by entities and is provided under a contract for a specific amount of time (de Roure, Jennings, & Shadbolt, 2003; Booth, et al., 2004). Formally defined, SOAs are:

> *"... a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains." (OASIS[10], 2004)*

SOAs are normally implemented through Web Services, although the two are technically different: an SOA is an architecture paradigm, and can be implemented in

[10] Website available at: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

practice in a number of different ways, whereas a web service is simply a mechanism for allowing access to a resource.

SOA allows for service orchestration i.e., the creation of complex web services through assembling simpler ones (Foster, Kesselman, Nick, & Tuecke, 2002). This is a powerful concept, because it means that a string of small services can be strung together, and packaged as part of a bigger service. Conversely, it also means that end-users could string together custom-made services, termed workflows. Workflows are attractive because they can be shared, edited and tweaked by others. In the context of e-science, sharing workflows is a useful method for passing around sets of operations or procedures. For instance, one scientist can build a workflow that makes a query on a dataset, and specifies that the results from the dataset are to be fed into data analysis code, then through a visualisation package, and finally for those results to be returned to some data storage. This workflow can be shared with a colleague, who can edit the workflow for their own purposes rather than writing one from scratch.

Whilst technically, web services and Grid technologies are different, fundamentally, they both try to address the same problem i.e., the management of distributed resources. Perhaps the most important difference to note from a broader perspective is that unlike Grid technologies, which have gained wide popularity in the academic world, web services have been developed in the business/commercial world. Thus, it is generally accepted that web services are a little more stable and mature in terms of technology and code development than Grid middleware.

However, as Atkinson et al. (2003) also point out, in addition to advances in computational knowledge, infrastructure will need engagement with other disciplines in order to make it "work". The emphasis here is upon the latter, where Atkinson et al. list some key challenges associated with the support for various applications of collaborative work, including scientific knowledge. They suggest that techniques are needed for a variety of purposes, mostly to do with understanding, supporting and promoting interactions between existing communities and the facilitation of new

forms of collaborative organisations (such as virtual organisations). David and Spence (2003) argue that whilst advances in technical infrastructure will create new scientific discoveries, "[i]n reality, these gains are likely to be the combined effect of social and technical transformations." Even supporters of large-scale technical investment admit that a gap exists between "raw capabilities" (such as bandwidth, storage and processor speed) and "realized [sic] performance" in terms of scientists' work (ibid). Thus, whilst one can be "reasonably confident" about the pace of technical advances, the greater uncertainty arises from whether benefits from those advances can be realised in practice.

2.5. Conclusion: The E-Science Vision

The e-science vision concerns the delivery of human and technical infrastructures to allow scientists to tackle larger, more complex research problems. Advocates of e-science highlight trends towards the need for more computational power in science, and the need to help scientists manage increasing volumes of data. They have also emphasised the need to build better and richer communication and collaboration tools in scientific communities. The promise was that by putting the computational infrastructures in place, it would enable scientists to access distributed resources, such as data, equipment or computational power, in an integrated way. This in turn, would enable scientists to collaborate with colleagues from different disciplines and different organisations on a routine basis. With this promise, the UK e-science programme was created as the first attempt to systematically develop e-science tools.

One critical aspect in realising e-science is the delivery of effective software applications and infrastructures to enable multi-disciplinary, multi-organisational collaborations. The delivery of these software tools and infrastructures will play a crucial part in the adoption and success of the e-science paradigm.

In light of the preceding discussions, understanding the requirements for e-science software will not be easy; on one hand, there is the need to use novel enabling technologies, while on the other, there is the need to support a set of evolving work practices. As a result, articulating requirements within e-science projects will be a complex negotiation between the 'problems' that the e-science technologies aim to address, and what possible combinations of solutions could solve those problems. In this context, it is important to find effective requirements engineering methods for e-science technologies, and to understand how requirements activities can be structured to facilitate the process.

Chapter 3: The Challenge of Articulating Requirements for E-Science Technologies

3.1. Introduction

Chapter 2 presented the background to the e-science vision. In this chapter, the focus shifts to considering software development work, and specifically requirements engineering work, in the construction of e-science technologies to realise that vision.

The chapter is presented in two parts. The first part addresses the issue of, and motivations behind the topic of requirements work in e-science projects. This part positions this thesis within the wider literature as attempting to provide a more in-depth strategic overview of the challenges that occur when articulating requirements and conducting requirements engineering work in e-science projects. The second part takes the investigation deeper. It maps out four factors derived from existing literature that might make e-science projects (and technologies) 'unique' compared to other software development efforts. These factors are (i) the characteristics of e-science projects, (ii) the population of end-users to be supported, (iii) the characteristics of the work the software has to support and (iv) the characteristics of enabling technologies. The purpose behind this mapping is to provide a more structured overview of the issues that scholars have pointed to. This map is returned

to in Chapter 8 (section 8.3) where it is enhanced with findings from the empirical work.

3.2. Evolution of Software Development Models

Software development has changed significantly in the past four decades. In the beginning, the development of software went hand-in-hand with the development of hardware. These software systems were discrete, standalone systems that only had to perform one or a few very well defined functions that the larger machine itself was built for. Since the days of large mainframe computers, computational devices have become cheaper and more affordable, and as a result have become ubiquitous pieces of technologies that can be found at work, home and on the move. The human computer interface has shifted from its early days with the communication with the computer directly through punchcards and tapes, to include the context and situation for which the computer is(Grudin, 1991). Thus, software systems need to: a) account for the growing contexts of use and b) perform a significantly larger number of operations and functions.

Models of software development have seen a number of evolutions and changes reflecting this trend. Often the impetus for change lies in finding more effective ways to make sure that the technologies being built are fit-for-purpose; i.e., ensuring the system 'solves' the problem that was envisioned. This is often preceded by discussions around how existing software development methods are inadequate to deal with the challenges of the ever-growing complexities that have to be incorporated inside these systems. At each stage of evolution of computers, one can argue in crude, general terms that there is a parallel revolution of software development models too. For instance, the move from single mainframe computers to networked desktop computing (especially in the workplace) prompted a move from the linear waterfall development model to a much more iterative spiral software

development model. In the latter, the system is presented to users at a number of limited points during development rather than simply at the end. The shift from desktop based computing to web based computing sparked agile software methodologies, which explicitly built in stakeholder feedback, continuously throughout the software development process. In the context of this thesis, the backdrop is that e-science technologies have been framed as large-scale, distributed, multi-disciplinary, multi-organisational systems. In light of this, the concern is that articulating requirements for e-science software will have to support an evolving set of working practices, whilst, at the same time, implementing the software using novel, bleeding-edge software for which there is little well-established knowledge or expertise to draw upon. There is a question of whether existing models of software development are adequate, or whether these systems are so complex that they merit new software development models of their own. If this is the case, what could those models look like?

3.3. Exposing the Complexities of E-Science Settings

In the literature a number of e-science researchers have raised questions about whether existing requirements engineering methodologies and techniques are sufficient for e-science. In particular, Zimmerman and Nardi (2006) argue that there are limitations of existing methods in understanding user requirements for cyberinfrastructures. They argue that requirements techniques have been developed for small-scale, standalone software, and thus are not suited to dealing with the challenges associated with requirements engineering for large-scale networked software systems, which support multi-organisational, multi-disciplinary collaborative work:

> *"Techniques of user-centered design and participatory design, while extremely useful, have been employed largely to design and develop small-scale, standalone systems under conditions of relative stability. They do not scale to the complexity and magnitude of*

> *cyberinfrastructures, nor can they keep pace with the speed at which these projects move."(Zimmerman & Nardi, 2006, p. 1604)*

In response to this a growing set of studies have presented some of the challenges gathered from empirical work, with a view towards creating a toolkit for future e-science practitioners to draw upon (see Warr (2007) for instance). Much of this work is based on the presumption that e-science projects have a particular uniqueness to them; a view that is supported in this thesis, and expanded upon in the next section. This work includes cross-case studies through interviews with participants in order to gather 'lessons learnt' that can be transferrable to other e-science projects.

This drive towards lessons learnt can, at best, be enlightening, and at worst, reduce the complexities of requirements and development work in this environment to simplistic terms. There are two dangers in the drive towards a toolkit approach, where practitioners are encouraged to read lists of best practices from which these lessons learnt have been derived. The first concern is that these recommendations have to be made generic enough for application across multiple settings. This can produce recommendations that seem obvious to a seasoned project manager without conveying the nuances involved in an e-science project. In this case, the result is still better than being without that knowledge; however, if it were possible to help future practitioners gain a more sophisticated understanding of how requirements are formed inside e-science projects, the outcome would be that the practitioner could better apply these lessons learnt. The second concern is that without a deeper understanding of the wider landscape (e.g., disciplinary structures, maturity of enabling technologies) that these projects are situated within, future e-science practitioners might lack the information needed to understand what works well in which context. Much of the argument in this thesis is based upon the idea that requirements are built up over time between project participants and the wider range of stakeholders involved in the process, as well as the landscape in which the project is set. Consequently, part of this work will involve setting out what characteristics exist in this environment.

Thus, this thesis attempts to address the gap in literature by providing more intricate details of the challenges in which requirements are articulated, formed and evolved during the development of e-science technologies. By doing so, it could help future practitioners to better understand the complexities that exist in e-science projects, which could lead to a better overview of the challenges inside the articulating and working out requirements for e-science technologies. Through these accounts, the analysis is taken further, by examining the characteristics of e-science projects, and mapping the issues and challenges that could raise problems in the requirements engineering process. This leads to the next section on mapping out the unique characteristics of e-science projects.

3.4. Part 2: Are E-Science Technologies a 'Unique' case?

3.4.1. Mapping what makes E-Science 'Unique'

In existing literature, e-science projects are often distinguished from other large-scale distributed software development projects. It is argued that it is particularly difficult to conduct requirements and design activities for e-science technologies, for a variety of reasons. Specifically, it is the combination of characteristics found within e-science projects that brings a 'unique' set of challenges to requirements work, compared to other similar distributed software development efforts.

In existing literature while there is no 'definitive list' as to what makes e-science projects unique, there seem to be a number of commonly agreed characteristics. This section provides an overview of the combination of characteristics that demonstrate the sorts of difficulties entailed in conducting requirements work in e-science projects. These factors are broadly classified into four characteristics as set out in Table 3.

Table 3: Challenges associated with conducting requirements work in e-science from existing literature

i) Characteristics of E-Science Projects	ii) Population of End-Users to be supported
• Geographically Distributed • Multiple and Diverse Stakeholders	• Heterogeneous Population of Users • High autonomy of End-Users
iii) Characteristics of the work the software has to facilitate	**iv) Characteristics of Technologies used to enable e-science**
• Supporting Scientific work • Supporting Cross-boundary work • Creating transformational changes	• Poor usability and immaturity of Grid technologies

3.4.2. Characteristics of E-Science Projects

It is argued that part of the challenge in an e-science project is its scale and distribution (Zimmerman & Faniel, 2006).

Due to the nature of the multi-organisational, multi-disciplinary collaborative work that e-science projects wish to support, they tend to have multiple and diverse stakeholders involved in the projects. As an illustration of this point, one of the case studies outlined later on in this research, TransProject, has ten Principal Investigators named with only seven members of full time staff.

It is not only that there is a high ratio between partner organisations (and thus, stakeholders) and staff members in e-science projects that is often a concern, but, as a number of researchers have noted (Zimmerman & Faniel, 2006; Zimmerman & Nardi, 2006), stakeholders tend to come from different organisations, or different knowledge domains, and thus have few overlapping concerns, and have their own

agendas and incentives for certain actions. This phenomenon is not new in requirements engineering; indeed, some requirements engineers have even argued that large-scale requirements analysis could benefit from the addition of 'political engineers' in order to manage the difficulties of requirements processes arising from the multiple interests of different stakeholders (Bergman, King, & Lyytinen, 2002a; Bergman, King, & Lyytinen, 2002b). Even in requirements and software engineering literature, however, there is no consensus on 'optimal' approaches to this problem.

Additionally, in e-science projects, the development team are often geographically distributed across the country, rather than colocated[11]. Geographically distributed software development processes are phenomena that are just beginning to be addressed in requirements engineering (Lloyd, Rosson, & Arthur, 2002); thus, there is no established expertise as to how such work can be done. In e-science literature, Momtahan and Martin (2002) note, that this distribution adds a level of communication to the project, as well as adding complexity, because consistency of project processes needs to be achieved across multiple organisations. Lee and Bietz (2009) make a similar observation, but note that the challenge is not to do with the distributedness of the groups, and more to do with tension in the coordination across organisations[12]. Lawrence (2006) in her research on e-research projects, also notes the difficulties in finding the right level of communication and engagement between project sites across distributed groups. If there is too much communication (e.g., emails), then project members might not read all of them, and if there is too little,

[11] Although as one can see from the following case studies, there tends to be a central base where there are more project team members based, the rest are often distributed across the other partnering organisations.

[12] However, e-science is not alone in tackling this challenge of coordinating distributed software development. It is a research area in its own right in software engineering. Plus, the coordination of requirements engineering efforts across geographical distances is also a research area in requirements engineering.

then there is the potential risk of information not being communicated to the relevant parties.

Also, agreeing on how to satisfy requirements becomes a lot trickier with multiple and diverse stakeholders. Momtahan and Martin (2002), who were involved in the European Union DataGrid project argue that different stakeholders will have different rewards, gains and motives for doing particular things on the project. Hence, with a large number of stakeholders, the number of requirements will also be large, because there will be differing needs for different stakeholders. This can lead to disagreements regarding what the project aims, objectives and deliverables are, which some scholars have argued leads to a level of ambiguity in the project (Warr, de la Flor, Jirotka, & Lloyd, 2007).

3.4.3. Population of End-Users to be supported by E-Science Technologies

One of the most noted challenges in designing e-science technologies is that the user population is often heterogeneous; different scientists use a variety of techniques and methodologies in the course of their research, which makes it difficult to define commonalities. They also use a variety of computing platforms, work in different organisations and engage with different research problems. As Zimmerman and Faniel (2006) summarise "the challenge in doing so [developing cyberinfrastructure] is that the users of such systems are numerous (potentially thousands), widely distributed, and heterogeneous in terms of culture, skills, knowledge, computing platforms, and other factors, all of which make it difficult to engage them in design activities". One way in which requirements engineering and other usability literature has dealt with this problem is through the use of user 'personas' as a proxy to represent multiple users; such 'personas' are, however, often employed in product-based development, where the range of users are fuzzy to define, rather than systems development, where the user population is generally easier to identify because users work within a particular setting or organisation. This has lead to suggestions in the

literature that requirements for e-science technologies are much more akin to product development than to systems development (Lloyd & Simpson, 2005).

At times this issue is compounded because participants in an e-science project come from different knowledge domains, especially in cases where e-science technologies are built to support cross-disciplinary work. For instance, NeuroGrid, an e-science project that aimed to bring together a number of experts on the domain of schizophrenia, was a working partnership between health practitioners, researchers and ontologists (Ure, et al., 2009b). The online portal that the NeuroGrid team had to develop needed to account for this diverse set of domain experts, each using the portal for their own purposes, and hence,each with a set of individual needs. As one can see, participants themselves are not homogenous. In the worst case, Warr et al. (2007) mention that in the Integrative Biology project, there was not a clearly defined community of users because the project aimed to support new cross-disciplinary work. Consequently, the project team found it difficult to elicit requirements when it was not clear whom they should be engaging with.

Another consideration in designing for scientists is that they are often given a high level of autonomy in their work and this has important implications for the adoption and sustainability of technologies. New tools or applications are often adopted on a voluntary basis, as opposed to the mandatory one seen in most commercial settings[13] (c.f. McLaughlin, Rosen, Skinner, & Webster, 1999). Scientists are unlikely to invest time into learning new tools unless there is a clear and direct benefit for their research (Lee and Bietz, 2009).

[13] For instance, see McLauglin et al (2004).

3.4.4. Characteristics of the work to be supported by E-Science Technologies

Supporting Highly Specialised Domains

Scientists are primarily engaged in knowledge production work. As well as involving highly skilled work, scientific disciplines often have a culture of their own (Knorr-Cetina, 1999). Each discipline has its own set of terminologies, some of which may have particular definitions related to that field that might not be apparent to outsiders[14]. Establishing and eliciting requirements for such a highly specialist domain necessitates developers and scientists to find shared references to progress; participants need common landscape to ensure that they have a common understanding when referring to the same aspect of the technology or of the domain.

For instance, Gibson, et al., (2006) describe their innovative approach to designing an interactive Lab book; they used a design methodology called 'Making Tea' where they asked bioinformaticans to make a cup of tea as an analogy to help computer scientists to understand the experimental process in bioinformatics. On reflecting on the use of the 'Making Tea' approach, schraefel and Dix (2007) argued that taking an analogy based design proved advantageous, since none of the computer scientists involved had advanced knowledge (beyond A Level) of chemistry:

> *"Without this domain expertise, it became readily apparent that while the team could observe the lab environment and the types of activities taking place ... getting at the critical issues – when, how and why certain things were recorded and others were not – was out of reach. The reason for this inaccessibility was simple: chemistry is a highly expert, abstract process. It requires deep domain knowledge in order to design an experiment, understand the chemical processes involved, and those the chemist desired to emerge. Without this understanding however, task analysis for instance, of the experimental recording process could not take place, and without such a model against*

[14] Sociology of science literature describes the intricacies of knowledge production, as well as the process by which scientific knowledge come to be codified, believed and established; sociologists point to the complex and contingent nature of scientists and the process by which scientific knowledge is generated (see Latour, 1987; Lynch, 1997; Knorr-Cetina, 1999).

> *which to design, the likelihood of developing a successful artefact went close to nil". (schraefel & Dix, 2007)*

Projects such as myTea highlight the challenges that are associated with creating a shared landscape and language between scientists and computer scientists/developers that has an important role to play in eliciting requirements. Another approach would be to use prototypes, as they serve as effective artefacts offering the scientists a shared point to refer to when considering their needs. In this context, even 'bad' prototypes can be 'good' because a significant mismatch between the scientists' idea of what the system should be with the current prototype system, might invoke extensive feedback from the scientists. Ure et al. (2009b) coined this notion as "giving users something to hate."

Supporting Cross-boundary Work

The point regarding the challenges of supporting multi-organisational work has been mentioned several times elsewhere in this thesis in further detail. To quickly summarise, one of the aims of e-science is to enable virtual organisations: lightweight organisations to facilitate collaborative work. VOs are defined as ad hoc organisations formed to share data, resources or to collaborate, which could last months, weeks, days or even hours (Foster, Kesselman, & Tuecke, 2001).

VOs make e-science infrastructures difficult to design because, as Lee, Dourish and Mark (2006) note, virtual organisations have to exist on top of existing organisations. They have to account for multiple computing infrastructures, multiple types of practices and multiple organisations of work. The issue is that in technical terms at least working across multiple configurations of technical platforms makes sharing resources difficult (Atkinson et al., 2003). For instance, security and authentication exemplify the sorts of problems that e-science technologies will encounter (as outlined in Chapter 2). Whilst at high-levels of requirements analysis, seamless sharing of data and resources might seem highly desirable, it is not until

requirements move into a lower, more detailed level, that the complexity of such a requirement becomes apparent. It is also important to remember that cross-organisational collaborations did exist prior to e-science; however, the effort and work associated with setting up cross-boundary work was costly. Thus, this was a barrier for more 'ad-hoc' inter-organisation collaborations to occur. The aim is to reduce the entry costs involved so that VOs can be routinely and easily created.

Creating Transformational Changes

Perhaps one of the more challenging characteristics to define in this section is regarding the desire for the e-science projects to realise 'transformational changes' in the scientists' work. As hinted in Chapter 2, the aim of e-science is to enable working practices that are a radical departure from existing ones, hence the potential need for significant changes in the social and cultural aspects of scientific work. In relation to requirements activities for e-science projects, this issue has serious implications for the selection of techniques to employ to elicit requirements. As Carusi and Jirtoka (2006) summarise:

> *"Getting users to describe the functionality of a system that does not exist, and that they do not know, is fraught with difficulties. These relate to the complexity of generating frameworks for users to understand what is possible of this new technology, how it can relate to their working practices and what potential benefit it may have on their organisational and business practices"*

If that is the case, then not only do end users need to develop an understanding of the technologies available, but potentially, new requirements techniques might need to be developed.

3.4.5. Technologies used to enable E-Science

Grid technologies were hailed at the start of the e-science programme as the technologies to enable e-science; however, Grid middleware at that time was still in its relative infancy, and scholars have noted its immaturity and lack of usability

(Chin & Coveney, 2004; Gentzsch, 2006). Chin and Coveney (2004), in detailing their experiences of using Grid middleware in RealityGrid and TeraGrid outlined the problems they encountered using the software. They criticised the poor documentation for the middleware (one of the most cited difficulties in the case studies presented later on), the large and "cumbersome" nature of the middleware resulting in the inability to build "lightweight" prototypes, and the configuration and administrative work involved in deploying middleware. Perhaps the most damaging (and principal) critique from the e-science development community is that they found the learning curve in using Grid components to be high for the computer scientists (ibid). Their argument is that if the developers found the software difficult to work with, then the learning process would be even harder for scientists, and thus have serious implications for the adoption and sustainability of the infrastructures. Interestingly, whilst there is little existing literature detailing the generic difficulties associated with building applications using Grid middleware, it is clear from the empirical work outlined later on that the number of bugs and lack of documentation for the middleware software was one of the most significant challenges in developing e-science technologies.

3.4.6. Structuring Requirements Work in E-Science Projects

In summary, conducting requirements work in e-science projects is challenging because it is characterised by uncertainty, complexity and novelty of the project itself, and of the technology that has to be built. As detailed above, e-science projects tend to be distributed across a number of different sites which make eliciting requirements difficult; they tend to have a number of diverse stakeholders, each with their own motivations and incentives for certain actions. The e-science software produced has to take account of a heterogeneous set of end-users, each with a high-level of autonomy, and thus has to account for their applications and infrastructures to be used on a variety of computing platforms, for a plethora of uses. The software also has to support highly skilled domain-specific work that has its own set of jargon

and terminologies, support collaborative work across multiple organisations, and enable and support new and 'transformational' changes in working practices of scientists. Finally, the software also has to be implemented using novel technologies for which there is a small user-population.

Despite these on-going efforts in working out requirements and design methods that can overcome the difficulties associated with e-science projects, there has been a significant lack of studies detailing how requirements work is currently carried out in existing projects. The research in this thesis aims to contribute to the existing literature, by conducting detailed studies of requirements work in e-science projects from the perspective of the team members tasked with the development of e-science software. The reason for taking this empirical approach to the issue is explained in the next section.

3.5. Summary

The purpose of this chapter has been to explain and justify the importance of the research problem being investigated (requirements activities in e-science projects).

This chapter started with a review of the existing literature on the requirements and design of e-science technologies. In particular, the emphasis was placed on separating and clarifying the 'unique' combination of issues which make e-science technologies difficult to conduct using 'traditional' requirements engineering practices. These issues include (i) the characteristics of the e-science projects, (ii) the population of end-users to be supported, (iii) the characteristics of the work the software has to enable and (iv) the technologies used to enable e-science. This was followed by a discussion on whether new requirements techniques are needed, or whether e-science practitioners simply need to apply knowledge and expertise from related disciplines, such as software engineering and participatory design.

In the next chapter, the discussion revolves around characterising e-science technologies and projects, to examine why requirements for such technologies are expected to be challenging, in light of the requirements literature in this chapter.

Chapter 4: Research Methodology

4.1. Introduction

This chapter details the research methodology undertaken in this research.

4.2. Empirical Accounts of Requirements Work in E-Science

One barrier to understanding how to effectively structure requirements activities for e-science technologies is the lack of empirical studies on the subject. Several researchers have argued for the need for more empirical research on existing practices (David & Spence, 2003; Edwards, Jackson, Bowker, & Knobel, 2007; Fry & Thelwall, 2006; Lawerence K. A., 2006). For instance, Edwards et al. (2007) points to the 'limited number' of studies on existing cyberinfrastructures carried out to date. They suggest that these studies are important because by learning from the experiences of project teams in existing projects, it is possible to inform and improve the planning and structure of future cyberinfrastructure projects. In particular, the lack of studies on requirements work in e-science projects has meant that the experiences and 'lessons learnt' from existing project teams have not been analysed and codified. The lack of feedback is worrying, since there is the danger that the same 'mistakes' will be made over and over again, and the knowledge produced from previous projects related to managing e-science projects will not be passed on.

As a result, this research contributes to the existing literature by taking an empirical approach and studying how requirements work in e-science projects is carried out in

practice. It is anticipated here that having this insight would provide future practitioners access to knowledge about how these technologies can be produced in a more systematic manner, hence reducing the time and resources involved in managing such an effort.

Of specific interest, the research in this thesis has focused on examining the practical issues that e-science team members are faced with, and how those challenges are dealt with on a pragmatic, everyday level in existing projects. In light of the literature review at the start of this chapter, if e-science projects are characterised as 'unique' and distinct from other large-scale, distributed software development projects, then *how are these challenges dealt with? Have e-science project teams found strategies to cope with some of the challenges listed above? Are there any particular issues that future e-science teams should anticipate and plan for? Do the e-science teams draw upon recognised requirements engineering techniques and if so, when?*

In the context of e-science and requirements engineering, this subject is of particular interest because there seem to be significant differences between the traditional view of software development, which begins with eliciting requirements, then through to design and then implementation, and the characterisation of the uncertain, volatile and continually evolving nature of requirements for e-science technologies that most scholars have pointed to (Voss, Procter, Jirotka, & Rodden, 2007; Momtahan & Martin, 2002). Thus, the argument here is that there are substantial insights to be gained by examining the 'nitty-gritty' details of requirements work in e-science projects, and detailing the very 'real' ways in which these issues are grappled with and worked out in order to inform future practices.

4.3. Case Studies Approach to Examine Current Practices

It has been argued, that the case study method is most appropriate when studying a contemporary phenomenon which cannot be studied outside its 'natural setting' and

real life context (Yin, 2002; Davidson, 2002; Benbasat, Goldstein, & Mead, 1987). Yin (2002) states that the case study method should be used when "the *phenomenon under study* is not readily distinguishable from its *context*" (emphasis added). Benbasat, Goldstein and Mead (1987) propose a similar definition of the case study strategy[15], but they go further, and state that there are three reasons explaining the use of case study research in Information Systems:

> *"First, the researcher can study information systems in a natural setting, learn about the state of the art, and generate theories from practice. Second, the case method allows the research ... to understand the nature and complexity of the processes taking place ... Third, a case approach is an appropriate way to research an area in which few previous studies have been carried out." (Benbasat, Goldstein, & Mead, 1987, p. 370)*

Thus, three important characteristics of case studies emerge – a) the phenomenon under study can be understood *in situ*, b) the method can enable the researcher to understand the complexity of the phenomenon, and c) it is most useful when the studied phenomenon has not been thoroughly examined beforehand[16]. My argument is that the case study research strategy was the most appropriate method for this research because the phenomenon (i.e. the requirements engineering process in e-science) meets the above criteria as argued in section 4.3: the phenomenon needed to be studied in situ; using case studies allows the richness of the requirements engineering process to be captured; requirements activities in e-science have not been studied thoroughly in previous literature.

---

[15] *"A case study examines a phenomenon in its natural setting, employing multiple methods of data collection to gather information from one or a few entities (people, groups, or organizations [sic]."* (Benbasat, Goldstein, & Mead, 1987, p. 370)

[16] Although some researchers have argued that case study methods would benefit from fields that have prior theoretical knowledge to draw upon (see Yin, 2002; 13). This is often the case in regards to fields that have a strong established theoretical base, and thus could utilise case study methods in hypothesis generation and in data analysis.

Before I outline the cases selected, I want to elaborate on the second point regarding the need to capture the richness of the requirements activities. One important aspect of this research is to understand the complexities that arise in the work of software and requirements engineers, in particular, regarding the social aspects relating to their work. For instance, Davis and Hickey (2002), whilst calling for more empirical studies of current practice warn that such studies should be taken seriously and not conducted in a superficial manner. In extending their analogy of why RE researchers must practice what they preach, they warn that understanding the RE practitioners' work requires in-depth study for a true appreciation. Similar to RE researchers preaching that an understanding of the end-users domain is of paramount importance, Davis and Hickey argue that studies of current practices have to "feel" the real work of practitioners:

> *"Note that studying current practice is not just doing a survey, any more than performing requirements analysis is just doing a survey. Analysis is all about thinking, experiencing, observing, listening, feeling, understanding pains, and so on." (Davis & Hickey, 2002, p. 110)*

Davis and Hickey's point is not simply that survey methods are inadequate, but that the RE process *deserves* more. They argue that the subject needs to be treated as a multifaceted entity because it is undeniable that requirements engineering (and software development in general) is a social process (Dittrich, 2002). One needs to understand and examine the real world work of developers to appreciate the real world contingencies that arise during this process (Monteiro & Svanæs, 1993). People having different perspectives, problems being exaggerated and tensions arising are all part of the rich backdrop that systems development exists in. All these constraints and pressures in the workplace will shape the final socio-technical artefact that is produced (see Williams and Edge (1996)).

4.4. Selecting Cases

Case selection is a crucial part of the research process. Depending on the research question, the type of research conducted, the availability of phenomena to study and the current knowledge available in that particular field, there are various methods for selecting cases. As with all case study research, there are two variables to decide: *should a single or multiple case study be undertaken?* and *which cases should be selected?* The two are obviously related (even if they are presented as separate issues here) since each case will be chosen dependent on its particular purpose within the research[17].

Three e-science projects were chosen:

- **First case study** –- *AstroGrid*. The UK's implementation of the virtual observatory.
- **Second case study** - *eMinerals*. To create a new paradigm of work in computational sciences that facilitates large scale complex simulations through collaboration and access to resources using Grid technologies.
- **Third case study** - *TransProject*. The building of the IT infrastructure for translational research in a city hospital

Multi-case study

For this study, I decided to take a multiple-case study approach; investigating three e-science projects. This was done for a number of reasons. Firstly, because there was not a strong theoretical base of studies regarding RE within e-science, this research

[17] For example, an *extreme or unique case* could be used where the study is atypical of the phenomenon under study, and thus, would either provide a "if it can work here it will work anywhere" scenario (Robson, Real World Research, 2002, p. 182), or a *unique* case that is worth studying purely for its rarity.

was of an exploratory nature rather than one that tested or proved a strict hypothesis. As a result, it was useful to study across multiple projects (rather than a single project) because there might be variations across the projects that were not anticipated. Secondly, common issues and patterns were the primary points of interest, and as such, findings across multiple cases would add weight to the research[18]. As Yin (1994) points out, evidence from multiple cases tends to be more compelling than a single case study. However, commonality is not the only point of interest. Thirdly, in situations where findings from one case were significantly different from either of the other two cases, or from expected or predicted results, this provides an interesting point of study. At that point, the concern is with the reasons behind the variation: why was there a difference? Was this deliberate? Or was there some characteristic regarding the situation which can explain the variation? One pitfall in this approach is that aspects of the studies may not always be directly comparable; however, the point here is not to make direct comparisons across the projects, but rather to seek the common or divergent issues.

Selection Criteria

In order to choose the case studies to examine, it was important to understand what attributes each case study possesses, and as a result, make choices about what cases would support the central research question. Ultimately, the goal was to narrow down to a suitable list of projects from which the case studies should be selected.

---

[18] To a certain extent, having a multi-case study is analogous to using triangulation across the studies; where triangulation is the use of multiple methods to add rigour to the research methods employed (Robson, Real World Research, 2002). Triangulation is traditionally associated with data collection methods, where a combination of different methods are used (such as interviews and surveys) in order to provide a 'fuller picture' of the phenomenon under study (Robson, Real World Research, 2002, p. 293).

One of the problems with identifying which projects to select is that the number of e-science projects is diverse. E-science projects varied in a number of characteristics[19] including size (some had one staff member, others had more than 20); duration (from a few months, to six or seven years); purpose (from proof of concept, to attempts at building long-term infrastructure); discipline (from astronomy to biology). They range from long-term projects in astronomy (AstroGRID) to relatively short-term projects in digital mammography (eDiaMoND). There are a number of differing characteristics which includes:

Time span – ranges from short-term prototypes to long-term infrastructure

- **Geography** – could be centralised or geographically distributed
- **Subject/Science area** – from astrophysics to medicine
- **Computationally intensive or Data intensive** – whether they rely heavily on computational power or heavily on data storage
- **Maturity** – completed, in the process of development, or beginning development
- **Middleware development or specific system/tool development** – whether the project is to build generic grid tools or develop specific tools for one specific project
- **Number of collaborations** – the number of collaborative partners there are in the e-science project

[19] There were two classifications which helped in drawing the distinctions between the different types of e-science systems – the first being David and Spence's taxonomy which classified e-science projects according to their purpose through four categories: community, data, computation and interaction. The second classification system which helped was the mapping available on the National e-science Centre website (NeSC). NeSC mapped the projects according to two dimensions – maturity of the code (from initial research to commercial code) and according to the level of component (basic infrastructure, general services and application components). The purpose of these classification schemes was not to produce a concise taxonomy of e-science projects, but rather to provide a rough idea of the landscape of e-science projects.

- **Open or closed** – whether it seeks contribution from a community or just a selected project team
- **Project size** – the number of people working on the project

The above list is not exhaustive. It is easy to see how the selection of e-science projects could be difficult with the number of characteristics that can differ. There were two major characteristics that the case study selection was based upon.

The first was the discipline from which the case studies should come. The choice was either to select three projects from different disciplines or select projects from the same discipline; for instance, all from the medical research domain. One significant advantage to the former scenario is that the data would be less biased towards the characteristics of one particular discipline. For example, data protection and patient confidentiality is highly valued in medical research and would be an integral part of any set of medical case studies, but such issues are not reflective of all e-science projects (especially those where patient data is not held). On the other hand, a same-domain set would mean that the projects are more likely to be comparable with each other. The choice was made to select projects from different disciplines (astronomy, computational chemistry, translational research), as one interesting question was whether discipline made a difference, and if it did what type of difference there was during the requirements engineering process.

The second was the maturity of the projects. The choice was to select projects that were at different stages of progression (e.g. beginning, middle or end), or to select projects that were all in similar stages. Of particular interest in this research was how requirements change and evolve over the course of an e-science project, plus whether the requirements engineering process changed as the project progressed. This included questions such as what strategies/techniques were used and were they effective? Was requirements engineering treated as a one-time phase at the start of the project? If not, how did the requirements and design of the infrastructure evolve? Since the first case study was based on the initial year of an e-science project, we

decided to pick two other projects that were mid-to-end project in order to provide a good contrast, and to allow us the opportunity to compare the data, to attempt to answer some of the questions listed above.

Following the above criteria, a shortlist was drawn up of potential case studies, and descriptions of the case studies were collated from the project websites and from published papers from the projects. This additional information was considered before making final decisions about case studies, as well as negotiating access. It is worth nothing that the projects were not all chosen concurrently. The first case study was chosen at the start of the research, with some initial data analysis conducted, whilst the second and third cases were chosen in order to complement the set of studies chosen.

Case Study 1 -*TransProject*

This case study was chosen as it provided an opportunity to study the development of an e-science project from its relative infancy. It provided a place for rich, ethnographic/observational data due to its central location and the relatively small development team. The project had a challenging remit of linking medical data, which traditionally has been seen as a difficult task in regulatory and organisational terms. In addition, the relative ease in gaining access to the case study was also a factor.

Case Study 2 - *AstroGrid*

For the second case study, it was decided that a more mature project should be examined, as this would balance the relative infancy of the first case study, and point out any issues relating to requirements during the deployment stages. In addition, the focus was to study a project that was not in the medical sector as the primary discipline of the project might influence requirements activities; therefore, the

difference in discipline may highlight particular characteristics. In addition, the scale of the project also made AstroGrid a good candidate for the research since the first case study had a relatively small team of seven people (cf. 23+ in AstroGrid).

Case Study 3 - *eMinerals*

In order to complement the other two case studies, the final case study was chosen to fit around them. In terms of scale, it was desirable for the final project to be near the middle of TransProject and AstroGrid, to acquire more breadth across the research. Again, in terms of discipline, the decision was to choose a project with a different primary discipline in order to attempt to highlight any differences that could be the result of the culture within that particular science. Lastly, an important factor in selection was the technology used in the project;: the previous studies did not utilise Grid technologies to a great extent (a characteristic discussed in the data chapters); therefore, studying a project which did use Grid technologies was deliberate to draw

Project	**TransProject**	**AstroGrid**	**eMinerals**
Size (in project members)	7	23.5	12
Discipline	Translational science	Astronomy	Computational Chemistry
Geographic distribution	Centrally located	Distributed over 5 sites	Distributed over 5 sites
Primary Choice of Technology	Combination of Java and other open source tech	Combination of Java and other open source tech, including Web Services	Grid technologies
Maturity of the Project (at time of study)	Early (3 months – 1 Year of 6 funded years)	Mid to end-term (5th year out of 6)	Mid to end-term (5th year out of 6)

Table 4: Comparison of all Three Case Studies (at the time of study)

out any issues relating the Grid, especially because it was widely seen as the enabling technology for e-science.

4.5. Data Collection

Primarily (although this might not always be the case), the case study research method has been associated with *qualitative*, as opposed to *quantitative* research. It is natural then, for data collection instruments within case studies to be of a qualitative nature; the most frequently used being observational/ethnographic methods, interviews (structure/questionnaire, semi-structure and unstructured) and document analysis. All three of these were used over the course of collecting data for this research. Part of the appeal of the case study research method is that because context is a significant part of the study, it provides rich, context specific data[20] (Yin, 2002).

A wide variety of data collection methods exist, including documentation, archive records, interviews, direct observation, participant-observation, films, photographs, videotapes, ethnographies, life histories (Yin R. K., 1994) (for a further explanation of the methods, see Robson (2002) or Yin (1994)). No single method is better than the others – each has their own advantage and disadvantage. Instead, as Robson (2002, p. 224) points out, the decision to use particular methods over others is

[20] In addition, part of the strength of using a case study strategy is its ability to utilise multiple sources of evidence through different data collection methods (Yin, 1994). Yin actively encourages case study researchers to use multiple methods; he argues that by doing so, the researcher can address more issues within the phenomenon and gain a more in-depth and rich understanding of it. But more importantly, that by using multiple methods, it allows for triangulation through drawing the same findings/conclusions through multiple sources: "The use of multiple sources of evidence in case studies allows an investigator to address a broader range of historical, attitudinal, and behavioural issues. However, the most important advantage presented by using multiple sources of evidence is the development of converging lines of inquiry ... Thus any finding or conclusion in a case study is likely to be much more convincing and accurate if it is based on several different sources of information" (Yin, 1994, p. 92) Thus, multiple data collection methods can complement each other in order to provide a richer sense of the phenomenon.

dependent on the rationale behind the choice, its practicality and what methods are available.

Data collection methods for the case studies were through a combination of observational methods, semi-structured interviews and documents. Each of the case studies had an element of all three methods – but some methods were used more than others depending on the case; the strengths and weaknesses of each method are discussed later on.

Observational data was the primary data collection method in the first case study. This was because there were few studies of e-science projects towards the start of the research, and carrying out observational work provided an opportunity to study the daily work of a project team, and how they grappled with eliciting and agreeing requirements. It helped to study the methods they used, and what issues arose over the course of the project. This provided the base upon which the other two case studies could be built.

With the second and third cases, it became obvious that observational methods would not be feasible due to the distributed nature of both of the cases examined. Thus, the primary data collection methods were through semi-structured interviews with team members including the Principal Investigators and the Project Managers. This was supplemented with documents such as papers, and the project wiki, which provided documentation and notes on the work in the projects. As well as interviewing members of AstroGrid, there were also small elements of observational work in these two projects, with observations made at an Access Grid meeting, and at a two-day user workshop. A summary of the data collection methods used for each case study is presented in Table 2. Methods for each of the cases are discussed in more detail in each chapter.

Case Study	**Data Collection**

TransProject	2 days per week over 10 months of observational work. 2 semi-structured interviews with the Project Manager and database curator, collection of 16 sets of minutes, including eight recorded weekly meetings.
AstroGrid	13 semi-structured interviews conducted with team members. On average the interview lasted around 1 hour and 15 mins with the shortest being 45 mins and the longest 1 hour 30 mins. Also, attended a two day workshop with users.
eMinerals	10 semi-structured interviews conducted with team members. On average, the interviews lasted around 1 hour, with the longest being 1 hour 35 mins, and the shortest 25 mins. Also, attended the Scotland Computational Chemistry Symposium in 2007.

Table 5: Data collection methods

Observational Methods

Observation is a qualitative data collection instrument in which data is gathered by the immersion of the researcher in the situation/environment under study. The researcher aims to gather data from the environment in which variables cannot be controlled. This type of research involves the study of the subject under natural conditions (Burgess, 1984). In its simplest terms, observation is made up of three elements – observing actions, listening to talk and clarifying through asking participants (project members) questions (Gillham, 2000).

Its advantage lies within the directness of the method – the data is collected through first-hand experience of the events and phenomenon rather than through what has been written on the topic, or what people have said on the topic (Gillham, 2000; Robson, Real World Research, 2002). As Robson (1993) notes:

> *"Observation also seems to be pre-eminently the appropriate technique for getting at 'real life' in the 'real world' ... but direct observation in the field permits a lack of artificiality which is all too rare with other techniques". (p191)*

It is this lack of artificiality that makes their approach appealing as a data collection method. In contrast with interviews, which are often criticised for bias arising from "deficiencies in memory" or due to the subject's desire to "present oneself in a favourable light" (Robson, Real World Research, 2002), observation can allow the researcher to examine the phenomenon in its *natural* state. Not only can the researcher observe and listen first hand to events unfolding, but observation of the actual situation provides the researcher with the ability to capture very rich detailed data. This richness provides the context in which the participants understand and make sense of the events as they unfold. In addition, as Burgess (1984) points out, this gives the researcher the opportunity to "obtain accounts of situations in the participant's own language which gives access to the concepts that are used in everyday life. The researcher can, therefore, construct an account of a social situation on the basis of the various accounts that are obtained from the informants."

Observation is not without its pitfalls. There has been research to suggest differences between what people observe and what is reported. Observation can be biased (see below) and can be "both fallible and highly selective" (Gillham, 2000, p. 47). Another major issue is the question of whether the researcher (as the research instrument) would change the phenomenon that they are observing (Gillham, 2000). This concerns whether the researcher themselves, simply by being there, would affect the phenomenon, so that inaccurate or behavioural changes are exhibited. However, as Robson (1993) adds, one cannot know what the behaviour would be if it was not observed in the first place. There can be no definitive solution, especially since one can never be sure whether a research instrument (in this case, the researcher) will ever affect the subjects under research. As Gillham suggests, one can only bear in mind the potential issue and make sound decisions about how the method is used:

> *"In real-world research ... the researcher is the research instrument, and any instrument used makes some contribution, has some effect on what is found. You have to make a*

> *consistent effort to observe yourself and the effects you might be having. ... A conscious attempt at rigour can usually lead to a reasonable judgement: we can expect no more" (Gillham, 2000, p. 47)*

In addition, one difficulty suffered with using observation within software development studies is that the work of programmers tends to be done alone, and as such it can be difficult to find out the rationale behind their work (D'Adderio, 2004).

On a practical level, observation is very time-consuming. The phenomenon, by the method's own definition, has to be observed in real time, it has to be written up into fieldnotes nd then coded and analysed (Gillham, 2000; Burgess, 1984). This, for example, would be slower than interviewing the same participants, or conducting a survey. Plus, fieldnotes and observation data are often more cumbersome to analyse than interview or survey data.

Since the researcher is the research method used, biases can be introduced and as Robson (1993) notes, acknowledging and understanding the types of biases that could creep into the research makes it easier to counteract them. He lists four types of biases:

- Selective attention. Since there is a lot of information to take in, a lot of the time, one pays selective attention within the setting. Robson suggests to "make a conscious effort to distribute your attention widely and evenly" (p204)
- Selective encoding. Expectations can affect the coding stage of the research - "try to start with an open mind-and keep it open."
- Selective Memory. The longer between the fieldwork and the fieldnotes being written, the more susceptible the recorded data to pre-existing expectations. "Write up fieldnotes into a narrative account promptly".
- Interpersonal Factors. The researcher might get to know some of the group better than others. "The general strategy is to seek to recognize and discount all biases".

Interviews

> *"The interview is a kind of conversation; a conversation with a purpose." (Robson, 1993, p. 228)*

Interviews are a way of eliciting information through getting the informant to talk about a certain subject. They involve speaking to the research participant, and asking them questions and recording the answers in some form (either on tape, paper, or electronically). It is one of the most commonly used methods in social research and is often used to complement other data in a multi-method strategy (Robson, 2002).

Interviews are commonly classified into three types[21]: *structured*, *semi-structured* and *unstructured*. Structured interviews have a pre-set list of questions, delivered in a particular order; they are considered as rigid research instruments (Burgess, 1984). They are often used in researching attitudes or feelings of a large research population.

In semi-structured interviews, the interviewer has a list of questions that he/she wants the interviewee to cover. These questions are generally used as an aide-memoir, and the order of the questions can be changed in order to adapt to how the interview is progressing. It also offers the researcher flexibility so that more time can be devoted to one topic, or the exact wording can be changed (Robson, 1993). Semi-structured interviews are the most important form in case study research (Gillham, 2000). Prior to using semi-structured interviews, there is a strong need to understand the key issues to be investigated in the research, and to determine which information is best obtained face-to-face, and which can be obtained through secondary means.

---

[21] Interviews can be classified in other ways such as formal and informal. In interview based research studies, all interviews are formal based, because they are all scheduled with the interviewee with a pre-agreed time and place, and probably recorded as well. In observational studies, informal interviews occur at any time, in any place during the workplace, and flow as conversations with research participants.

Unstructured interviews have even fewer restrictions. There are a number of topics (upon which the interviewer can agree with the interviewee at the start of the interview), and the interviewee can be free to discuss whichever topic they wish, in whichever way they wish. The conversation can develop and is completely informal (Robson, 2002).

Semi-structured and unstructured interviews are often used in multi-method research, especially to supplement participant observation. This is because they provide an ideal instrument to explore complex and ambiguous issues with those that are directly involved (Arskey & Knight, 1999; Robson, 2002). They allow the participant to provide a rich account of their actions, feelings and perspective upon the research subject. In the context of multi-method research, interviewing methods allow the observer to gain a better insight into the work involved in the workplace. The interviewees can explain their actions and rationale for establishing and agreeing requirements within their projects.

In addition, interviews can allow researchers to gain indirect 'access' to places which they cannot gain access to in reality because of time, situation or place. For example, they can give access to the biography of an individual (Burgess, 1984). In relation to the research in this thesis, examples include meetings or activities that are not witnessed by the researcher.

One of the main disadvantages in the semi-structured/unstructured interview method is that it is time-consuming. As Gillham points out, "richness comes with a price" (2000, p62). Each interview has to be arranged with the interviewee, rooms have to have booked, equipment checked and appointments reconfirmed. In addition, the notes have to be typed up and/or interviews have to be transcribed; these two steps alone take a significant amount of time. Data analysis can also be lengthy (Robson, 2002). In addition, the interviewee might provide a biased or inaccurate account, either deliberately or accidentally (Robson, 2002).

Documents/Wiki

Apart from fieldnotes and interviews, another major source of information is the use of documents (written or electronic). Documents are a "particular kind of artefact" (Robson, 1993, p. 272) that can be studied. They can include:

> *"... court records, minutes, contracts, letters, memoranda, notes, memoires, diaries, autobiographies and reports." (Burgess, 1984, p. 272)*

Or in some cases, even films, television programmes or drawings (Robson, 1993).

In the context of this research, the types of documents consulted were largely each project wiki[22], published papers from the projects, systems specifications, research proposals and, sometimes, emails. The availability of the published papers and research proposals has greatly improved due to the internet. These published papers complemented the interviews and observational accounts greatly.

The project wiki has provided a rich resource from which the researcher can understand the work within an e-science project. Largely due to the distributedness of e-science development work, team members often used the wiki in order to coordinate meetings or publish agendas/minutes, or even as their worklog[23]. A large repository of information existed in the wiki, and often included details and accounts that previously would have been difficult to obtain in such an open manner. Implementation and technical details could be written up as wiki pages alongside the rationale for a particular design.

However, as with all research instruments, documents and wikis should not be viewed uncritically. In the case of documents, they often provide tidied versions of meetings, events or research. It is possible that they could omit details that could

[22] Which all three projects had.

[23] This occurred in eMinerals where each of the project scientists kept an online worklog in order to allow other scientists to keep track of their progress without the obstructive nature of asking (constantly).

prove critical to the research. In addition, one disadvantage of the project wiki is that wiki pages tend to change over the course of a project so information could be lost; and generally speaking, information on the wiki could be inaccurate, or even incorrect if the page is outdated. In addition, sometimes it is not immediately clear who has made what changes to the wiki page, and therefore, traceability of the authorship could prove difficult. Despite these disadvantages, the project wiki still provides a great supplementary resource in understanding the requirements and software engineering processes.

4.6. Data Analysis

The method by which the data are analysed is crucial in understanding how conclusions are drawn (Silverman, 2002). Data analysis is driven by rigour and an appreciation of viewing alternative interpretations of the data presented (Yin R. K., 1994). Although the data analysis is presented as a linear process here, analysis was not conducted linearly but rather, in an iterative manner; often from analysis of individual cases, to cross case analysis and back.

Managing data analysis for three case studies is no trivial task. According to Eisenhardt (1989) there are two types of analysis that could occur in case studies: *within-case* and *cross-case. Within-case* analysis involves "detailed write-ups" for each case, with the purpose of being "intimately familiar" with each case on its own (ibid, p540). In this research, this was done through analysis of each case, and presenting the analyses in a descriptive manner. Common themes would emerge throughout both data collection and during analysis; these were noted down, and in the case of the first case study, fed into the questions and direction of study for the second and third cases. Of particular interest were instances of phenomena which led to conflict within the group, or issues that the team identified as having particular problems with.

The above step helped the second part of the analysis, which is *cross-case* analysis, in order to spot or highlight patterns across the cases. As Eisenhardt points out, conducting cross-case analysis attempts to counteract the tendencies to make claims based upon insufficient data or biased towards certain data (e.g. certain interview respondents). She also lists three tactics in order to deal with cross-case analysis, the first two of which were used in this research: a) examining a selection of categories and themes across the cases for similarities and differences; b) selecting two or three cases to compare for similarities and differences in order to "force" the researcher to examine the subtle differences between cases; c) examining the data according to the data source (e.g. interviews, surveys etc.) to cross-examine whether evidence from one data source substantiates claims from another.

In this research, the interviews were transcribed in full, and were coded according to open coding strategy, which does not use a predetermined set of codes, but instead, allows a set of relevant codes to emerge from the data and throughout the analysis. These codes provided the themes and categories from which to compare the same phenomenon across the cases. The first stage of analysis was to examine each project on its own, making sense of the requirements activities and the project as a whole from the various accounts. Then, findings from the three cases were compared, with particular attention being paid to similarities and differences across the cases (especially those which were not anticipated). Once these were noted, the analysis moved to explaining the context in which this occurred, and providing interpretations of the events. As noted above, alternative interpretations had to be considered during analysis. Throughout the analysis process, the concepts from the data were compared with existing literature, and by going back and forth between literature and themes, lead to further insights within each case and comparatively across the cases.

4.7. Summary

The purpose of this chapter was to outline the research methodology by which this research was carried out.

One barrier in understanding whether new techniques are needed is that there has been a lack of empirical data about how requirements analysis is currently dealt with in existing e-science projects, i.e., accounts of the practicalities of requirements work in relation to e-science. With a better grounding in how requirements work is currently conducted, and the issues and challenges that are faced from the perspective of e-science project team members, it is possible to build up a better picture of the sorts of requirements work that e-science team members engage with and to examine the way that requirements work is currently structured. By doing so, the aim is to move beyond simply theorising about the challenges of requirements work in e-science, and to shift the emphasis onto examining how requirements work could be better informed in the future. In other words, by conducting empirical studies of the requirements activities in e-science projects, it is possible to provide a significant contribution to more effective and systematic development of software to enable e-science.

To do so, three e-science projects were chosen as case studies in this research. A case study method was taken in order to capture the complexities of e-science, since constructing requirements is a social process. Having case studies would provide the contextualisation desired to gain a thorough understanding of how requirements were worked out, shaped, prioritised and influenced by a range of factors. A mixture of observational methods, semi-structured interviews and document analysis were used to gather data. To balance depth and variety/generalisations of the research, three case studies were undertaken. The rationale behind the selection of each case study is presented, and a discussion of the data collection and analysis is also presented.

The following three chapters describe the findings from each of the case studies conducted in this research: Chapter 5 presents the AstroGrid case study; Chapter 6 presents the eMinerals project and Chapter 7 presents the TransProject case study.

Chapter 5: AstroGrid

5.1. Introduction

Requirements work in an e-science context has often been described as uncertain and complex. The set of requirements is often characterised as volatile (Momtahan & Martin, 2002), as the novelty of the applications and infrastructures developed are emphasised. One interesting aspect is to consider how this uncertainty is managed through the organisation of development work in e-science infrastructures, and to consider how requirements activities fit into that organisation.

E-science technologies, by their very nature, are difficult to elicit requirements for because the working practices desired have to be reimagined in parallel with the technologies the participants are seeking to enable such practices. This elicitation process is also exasperated by two characteristics that are an inherent part of e-science projects. The first characteristic is that there is a wide and diverse pool of stakeholders that needs to be consulted. E-science projects are largely dependent upon several different organisations to support the infrastructure during and post-development. Such stakeholders can include those that are responsible for the development itself (in this case study, the Principal Investigators), data providers (the observatories), ordinary scientists (astronomers not involved in the project) as well as the software developers who are ultimately responsible for delivering the software that has to meet such requirements. The second characteristic of e-science projects that makes elicitation of requirements challenging, is that the working arrangements

that the software has to enable are expected to be significantly different compared to existing practices. Traditionally, user interviews form an integral part of good software development practice where there is a human user involved (cf. machine to machine). Often, there is a heavy reliance on the end-user to articulate their desires and needs, which in turn heavily influences the requirements being articulated. However, in the case of e-science projects where the resultant working practices are expected to be vastly different from existing practices, requirements elicitation is a much trickier task, as the traditional user interview approach is not as easily applicable here (Zimmerman and Nardi, 2006).

In light of these two issues, a recurring observation throughout this research was the need for project participants to continuously adapt their approaches to requirements elicitation, and even to use multiple elicitation techniques at the same time. Note that this is explored further in Chapter 7.

This chapter aims to show how the explicit development process used by the project helped to orientate team members in understanding the types of requirements and the granularity at which to consider requirements, over the course of the project. In doing so, it is possible to see how the complexities of requirements for a large-scale data infrastructure are managed on a practical basis. The concept of temporal patterns (Reddy & Dourish, 2002) is used in organising this chapter.

The AstroGrid project was one of the largest e-science projects in the UK E-Science programme. Its aim was to develop the UK's implementation of the Virtual Observatory (VOb), an online infrastructure which would allow astronomers to search, store and analyse multispectral astronomical data. The VOb's vision has a strong resonance with the e-science vision, i.e., the need to federate large datasets, to provide tools for astronomers to analyse data more effectively and to enable remote access to datasets. Consequently this is a project which has the philosophy and ethos of e-science at its core. Coupled with a history of large-scale collaborations in the astronomy community, the AstroGrid system provides a natural starting point for exploring requirements activities in the context of an e-science project.

As one would expect, working out the requirements for the AstroGrid project and developing the software to enable it was not a straightforward task. There were a number of complex socio-technical challenges to overcome in order to produce a working prototype that would be useful for the wider astronomy community. This chapter focuses on the requirements activities that occurred during the first five years of the project, detailing: the project team's initial approach to development, the development process they worked with, the way that requirements activities were structured and the way that team members were orientated to both requirements and development activities. In addition, I will also describe how certain types of requirements are emphasised as the project progressed, as well as the impact that the wider environment had upon the requirements for the system.

5.2. Background

5.2.1. Astronomy Work: A Brief Overview

Astronomy is the study of "celestial phenomena" (Roy & Clarke, 2003) that includes stars, planets, and galaxies and their physical properties. Because it is only possible to observe these phenomena, astronomy is an observational rather than an experimental science. Astronomers use physics to explain the objects that they observe. From optical, to radio waves through to electromagnetic radiation, astronomers use a variety of scientific instruments to measure the radiation given off by a variety of phenomena.

Astronomy has often been cited as one of the oldest sciences. Astronomers have traditionally been characterised as working alone, or in small groups or laboratories[24]

[24] Although this is not strictly true depending on the sub-discipline that one examines. For instance, as Galison and Hevly argue, radio astronomy, largely driven by physicists, has been working to "big science" efforts in the early 20th Century (Galison & Hevly, 1992).

(McCray, 2000; Galison & Hevly, 1992). After World War 2, post-war astronomy shaped to take a more coordinated approach in general. Astronomers began to use more sophisticated equipment to collect observations in other bandwidths and thus, created new sub-disciplines such as x-ray astronomy. As bigger and more accurate instruments were sought after, costs drove the need for collaborative work between multiple institutions. Consequently national and international alliances became commonplace. McCray (2000), in his sociological study of economics of large telescopes, outlines the fact that astronomy in the UK and Canada relied solely on Governmental funding (cf. U.S. where private funding played a part). International and national based alliances helped to alleviate costs and allowed sharing of equipment. One example of this type of "coalition building" in astronomy is the construction of the Hubble Space Telescope (Galison and Hevly, 1992, pp184-211), which cost two billion U.S. dollars, and required a coordinated effort across NASA[25] and ESA[26].

Computation and networked technologies have begun to play a significant role in astronomy. Advances in computation have allowed astronomers to create models and complex simulations to further test their theories. With particular theories in mind an astronomer could use models to test their hypotheses, and this in turn drove the demand for further computational power. The advances in computational techniques have also allowed astronomers to use more sophisticated data analysis techniques. Recently, the popularity of the internet has driven the use of networked technologies to share data and information, and most data centres/observatories will publish part of their dataset online.

---

[25] National Aeronautics and Space Administration

[26] European Space Agency

5.1.2. The Growing Need for a Systematic Approach: The Virtual Observatory Vision

By the late 1990s there were concerns raised in the astronomy community about the need for better data infrastructures and tools to exploit the large volumes of data being produced by the bigger instruments available (Djorgovski & Williams, Virtual Observatory: From Concept to Implementation, 2005). For instance, the Sloan Digital Sky Survey (SDSS) will map a quarter of the entire sky; it is expected that the final amount of data produced will be over 15 terabytes, a figure according to the SDSS, which would rival the Library of Congress (SDSS Website, accessed July 2007). It has been estimated that the volume of astronomical data is doubling every one and a half years and one terabyte of data is being produced every day (Djorgovski, 2005).

Advocates of the Virtual Observatory argued that without adequate measures being taken, the danger was that the datasets produced would be under-exploited because of the lack of tools and ability to do so. In particular, the concern was not only to do with the volume, but also to do with the heterogeneity and richness of the data available, as well as the federation of that data (AstroGrid, AstroGrid 2 Project Proposal, 2003). Some of these concerns were on a practical level resulting from working with large datasets, larger datasets would take significantly longer to open than the small datasets which astronomers had been used to, emergent patterns would not be immediately noticeable since it would be impractical to spot patterns purely by hand (or eye), and remote access to datasets would prove more difficult as the astronomer attempted to download terabytes of images[27] (see Szalay & Gray, 2006;

[27] In particular, accessing remote datasets was very labour intensive because astronomical data were published via static webpages by data centres (which are attached to the observatories). Firstly, an astronomer would need to find the right dataset by hand (i.e. via a website). Secondly they would have to download each image individually. Since much astronomy work is time-based, this process became time-consuming and rather monotonous as astronomers had to manually find and download a series of images by hand.

Szalay & Brunner, 1999; Djorgovski & Williams, 2005). Other concerns were to do with working across multiple large datasets: issues of interrogating data across multiple datasets and being able to search and combine data on a routine basis and in a standardised manner[28] (AG Report). The desire to conduct multi-archive science and large database science were cited as two of the main drivers for the Virtual Observatory (outlined in the next section). In addition, as mentioned before, the prevalence of the internet has driven the expectation of the astronomers to gain remote access to datasets and conduct more of their work online. The widespread use of internet technologies has driven demand as well as expectation that remote access and other supplementary services such as online storage[29] could be and should be easily done.

5.1.3. The Virtual Observatory

Recognition of the issues outlined above, i.e., the drive for a more coordinated approach to data federation and improved network technologies to facilitate astronomers' work, lead to the discussion of the Virtual Observatory (VOb).

The broad aim of VOb is to produce an infrastructure allowing the majority (if not all) of the astronomical data in the world to be made available to the astronomer through a single, simple-to-use interface:

---

[28] Another issue was the desire to conduct multispectral work. Astronomers tend to specialise in one spectrum, (e.g., radio astronomers or optical astronomers) and it is unusual for multispectral work to occur. This is not so much to do with problems with access to the other types of data, but more to do with resource constraints. Working with multi-spectral data requires specialised expertise, so unless there is a clear and direct benefit, there is little incentive to do such work.

[29] Online storage could include multi-organisational storage so that astronomers from different institutes can share data; the size could be terabytes, a scale that would be difficult using current technologies

> *"The Virtual Observatory will be a system that allows astronomers to interrogate multiple data centres in a seamless and transparent way, which provides new powerful analysis and visualization tools within that system, and which gives data centers a standard framework for publishing and delivering services using their data. This is made possible by standardization of data and metadata, by standardization of data exchange methods, and by the use of a registry, which lists available services and what can be done with them." (IVOA Website, accessed July 2007)*

The system would facilitate the comparison and integration of data from multiple data centres, across multiple spectra and provide intelligent tools to enable the astronomer to analyse the data to facilitate discoveries. The AstroGrid 2 project plan describes the metaphor of the VOb in terms of a real observatory:

> *"The world-wide super archive becomes the sky, and software the instrument with which we collect data from the sky – hence the metaphor of the Virtual Observatory" (p1)*

Besides the data federation that is clearly needed, the vision also requires an infrastructure that can deal with service orchestration (AstroGrid, AstroGrid 2 Project Proposal, 2003). It is argued that the VOb will be a set of standards and protocols that have to allow services and data to be interoperable (Szalay & Gray, 2006). In many ways, the VOb vision resonates with the e-science vision: finding methods to manage the federation of multiple, distributed online databases, facilitating multi-organisational work, and building the infrastructure and tools to enable astronomers to search, analyse and store that data. Plus, with a pre-existing history of coordination and collaborative efforts, astronomy and the VOb provided a flagship example in much of the UK E-science programme literature[30].

Before beginning to discuss the details of the case study in this chapter it is worth briefly mentioning the different projects that are involved in working on the VOb. Of

[30] AstroGrid as a flagship example of the UK E-science programme is evident in its being mentioned in a number of the 'visionary' papers in e-science; see Hey and Trefthen (2002).

most significance is the National Virtual Observatory (NVO) in the United States, which is also working on building an infrastructure for the VOb. Then there is the Astrophysical Virtual Observatory (AVO) in Europe, which focuses more on the scientific tools for the VOb, and partnered with the UK in the VOTech project, a sister project to AstroGrid that focuses on producing data mining techniques. The NVO, AVO and AstroGrid went on to become the founding members of the International Virtual Observatory Alliance (IVOA): a forum for the collaboration and cooperation of VOb efforts. In practical terms, the IVOA became the body for standardisation of data formats and protocols across the different Virtual Observatories. At the time of writing[31], there are 16 members of the IVOA.

5.3. AstroGrid: The UK's Virtual Observatory

5.3.1. The Project

AstroGrid is the UK's implementation of the Virtual Observatory. Launched in late 2001 with the second phase of e-science funding finished in 2007, AstroGrid is a joint collaboration between 11 universities across the UK. The project is one of the founding members of the IVOA and has a sister project called Virtual Observatory Technologies (VOTech) which focuses on techniques and algorithms for astronomical data. Funded by the PPARC E-Science funding allocation, the project eventually received £14 million of funding over these two phases, alongside a final operation phase (AstroGrid 3) that is not covered in this case study (AstroGrid Website[32], 2009).

---

[31] Aug 2009

[32] http://www.astrogrid.org/wiki/Home/AboutAstroGrid

The project vision is two-fold. The first aim is to produce software components to enable the VOb. The vision was to create an "open framework approach" allowing a mixture of different components to be assembled and used in a "plug-and-play" fashion (AstroGrid, 2001). In practical terms, the AstroGrid team would produce a series of standardised components that would fit into a grand generic software infrastructure. These components can be swapped in and out for other implementations of the same component (an example of this is the workflow engine, which is detailed later on in this chapter). The aim is to produce components with standardised interfaces, meaning that they can be re-used elsewhere, such as by other VOb projects. Plus, different components from other projects can be 'plugged' into the AstroGrid infrastructure, facilitating sharing of software across the different VOb projects. One interesting result of this emphasis on infrastructure is that the AstroGrid project will develop few applications for end-users. Instead, the focus is on developing the standards (e.g. data formats) and the software infrastructures for end-user focused applications to sit upon. By focusing on infrastructure development, the AstroGrid project has built up significant expertise in this area compared to other international VOb projects, which have tended to focus on applications. Secondly, AstroGrid aims to deliver a working system for UK astronomers. Their approach is to develop and deploy software components across the UK's data centres to develop a nation-wide infrastructure for astronomical data publishing. As well as the infrastructure, applications and tools to take advantage of the available data would be integrated into the AstroGrid client for astronomers to use. This aspect of the project created some tension, as some members of the project pointed out the different priorities between writing software for prototyping/research and writing software for deployment purposes (see Lawrence, 2006).

With 23.5 full time staff at one point (AstroGrid, 2003), this project is the biggest of all the case studies in this thesis. The project team comprised of two groups: software developers and project scientists. The developers were responsible for working out the requirements and building of software components. Driven by the vision of building a 'professional' product with proper software engineering practices, the

preference was to hire developers with industrial experience. This is unusual for an e-science project because most projects partner with a computer science department to develop the technologies. Each developer had a specific component that they were working on and were expected to move to a different component during each cycle, while later on each developer had an area of responsibility. In keeping with the professional development practices, this was done in order to build redundancy into the development process (Interview, AstroGrid Technical Lead, AG/3).

The project scientists' role was to provide the science input into the project. Five astronomers were partly funded by the project and they were selected to complement each other so that the team had expertise that covered different areas of astronomy, including solar physics, optical, radio and x-ray. The group was lead by the Lead Project Scientist, who shared the same role in the VOTech project.

The Project Manager dealt with day-to-day management, while the management of the developers was the responsibility of the Technical Lead. As the project progressed, the role of the Project Manager shifted to become the Programme Manager of both AstroGrid and VOTech because the major tasks of these two roles were to form links and foster partnerships, in order to enable the open framework infrastructure AstroGrid was working towards. During this shift, the Technical Lead became the Project Manager for the project. For clarity purposes, the original roles were used in describing the case study, partly because most of the interviews talked about the early part of the project, and their original roles provided a better representation of their roles in this early stage. Overall, the AstroGrid Lead Investigators (AGLI) Board was responsible for the management of the entire project. The board was comprised of the Principal Investigators from each of the partnering organisations.

5.3.2. The AstroGrid System

Astronomers can log in using an AstroGrid account and perform a range of tasks including searching for images, storing data, creating videos of images, and using visualisation tools to analyse data. For instance, an astronomer can specify which region of space they are interested in using coordinates and the radius, and the system will return all the data sources which have data in that region. Once the data is returned, the astronomer can store it in MySpace (a distributed online storage system built by the team). The astronomer can then set up a workflow[33] that can run analysis tools or visualisation tools with that data. The results of this workflow can be stored back onto MySpace, and this in turn can be downloaded onto the astronomer's local computer. In addition, a number of tools and plug-ins were developed that enabled astronomers to share data between different analysis tools. The initial interface was web-based, but the switch was made to using a desktop client (using java webstart) in order to allow for improved usability, e.g., better response times, and increased flexibility in visualising data.

On the broader, infrastructure level, the AstroGrid system has a multi-layered architecture. As mentioned previously, the system was built with a strong focus on component independence. Thus, the whole architecture is split into a series of components[34]. The core of the project's effort was focused on the Virtual Observatory Infrastructure layer and it is in this layer that most of the discipline specific components were based[35]. The major components built were the workflow

---

[33] Astronomers could also take workflow templates and make their own workflows to string tasks together; for instance, the retrieval of data and the basic analysis of that data.

[34] Some of the components were built in the first phase, and some work was done to replace some of these components in the second phase, depending on whether their functionality had to be improved or adapted to work with new components (AG Technical Scope).

[35] Components in the middleware layer were mostly seen as non-discipline specific, meaning that it is possible to swap in similar components from other e-science projects (although this might involve significant work).

engine, registry, MySpace (online storage), Dataset access and security, and community management.

It is also worth noting that the new VOb infrastructure had to sit upon existing organisations and infrastructures, and this aspect of work is something that was a recurring challenge for the AstroGrid team. Securing the support and access to resources had to be negotiated with existing data centres and universities. For instance, allowing access to a dataset would involve some work by the data centre involved in configuring database connections; this might even involve a server being hosted by the data centre. This was a crucial aspect of the AstroGrid project, as the value of the system will increase in proportion to the volume of data available, meaning that the goal is to work with as many data centres possible.

5.4. Building the Virtual Observatory: The Missing Middle

Despite having a history of working in large-scale international collaborative projects, the AstroGrid team found developing Virtual Observatory software to be a much more challenging prospect than previous efforts. The AstroGrid system represented a departure from the type of large-scale instrument building they were used to. One of the team members remarked that AstroGrid was clearly the largest coordinated astronomy software project undertaken (Interview, AstroGrid Developer, AG/6). They found that getting the 'right rhythm' in structuring their process was not straightforward, especially since the project was situated in an environment spanning across the UK astronomy community, other VOb projects and other e-science projects.

Another challenging aspect of this project was that no single Virtual Observatory had existed before. It was not clear what one should look like, or how it should be built; there were no pre-existing templates to draw upon. The project team had to work out the details of the VOb as they were building it. As the Lead PI described, there

existed a "missing middle" that the team had to grasp, i.e., the architecture and components to facilitate data sharing and reduce the "data complexity" (Interview, AstroGrid PI, AG/2). With this lack of knowledge, the developers often saw their work as research and development rather than simply writing robust, solidly-engineered software. With this slant towards research, user-engagement was an integral part of their work. However, this was not a straightforward matter, as the developers seemed to engage in different methods of requirements elicitation techniques dependent on the different points in the development process.

5.5. On Switching Requirements Techniques and Stretching their Utility

5.5.1. Traditional Requirements Elicitation

In the early stages of any software development project, the development team has to begin to define what is to be designed and developed. Part of this process will include requirements elicitation that will determine the context of use, the needs of end-users, and the conditions in which the system has to perform (Goguen & Linde, 1993).

Choosing an approach to requirements elicitation is dependent on a number of factors. Each requirements technique has its own advantages and disadvantages. Plus, their advantages are partly dependent on the scenario in which these techniques are employed, and also by whom (Davis, 1982). These factors relate to the general aspects of the project, for instance time and costs available. Others factors can include the sorts of insights that the requirements engineer can gain from a specific use of the tool; for instance, the use of prototypes can generate discussion, which can be useful in a context where requirements are challenging to clarify by stakeholders.

5.5.2. Requirements Elicitation in Infrastructure projects: Shifting demands on the techniques

It is well acknowledged that in e-science projects, in particular projects such as AstroGrid, where there is a substantial element of infrastructure construction attached to them, that project participants have to take a much longer term view of the systems they are constructing (Ribes and Finholt, 2009). Karasti et al., (2010) use the concepts of 'project time' and 'infrastructure time' to emphasise the different time frames that projects constructing large scale infrastructures exist in. They highlight how infrastructure projects are placed "toward an emergent future along an unfolding long-term path of collaborative infrastructure development" (Karasti et al., 2010; pp403).

This sense of collaborative development over a long period of time means that single phase requirements capture becomes inadequate over the lifetime of constructing an infrastructure. This evolution can also be seen through the empirical work. If infrastructures are conceptualised as evolving structures in relation to requirements articulation, one interesting way in which this manifests itself is by how requirements elicitation becomes part of an on-going process, so that multiple techniques become employed over the course of a single project. Particular techniques serve particular needs of the project depending on its phase. For instance, in the detailed example below, the science use case was a popular way of expressing the sorts of work that the AstroGrid team sought to enable at the start of the project: it provided a clear way of demonstrating to the developers who had little domain knowledge, as well as to the outside world, what problems AstroGrid could solve. However, as the priorities of a project move on, the needs / demands of the requirements techniques also change. Interestingly, as this next case study shows, sometimes, it can take a while before it is recognised by team members that the demands on the requirements techniques have shifted, and project participants attempt to make the existing technique 'work' before switching to another one. Inspired by the idea of each technique having a particular usefulness to each of them (the techniques) the next

section details the account of how requirements techniques can be stretched over the course of an e-science project's progression.

5.5.3. Stretching the Utility of Requirements Techniques: The example of 'Science Use Cases'

For the AstroGrid project participants, they recognised that an integral part at the start of the project would be to set out the scientific direction of the project. Thus, one of the earliest tasks in the AstroGrid project was the production of ten science cases, encompassing the variety of scientific interests in the consortium tasked with building the VOb. These science cases, which came to be known as the AstroGrid Top 10, were seen to capture the scientific requirements right at the start of the project, and covered the majority of subdisciplines which existed in astronomy: from radio, x-ray, image and spectral astronomy (Walton, Lawerence, & Linde, 2004). They were expected to "represent a cross section of current topical astronomy, solar and STP research areas with functionalities covering a wide spread of technical areas" (ibid). According to one AstroGrid paper describing the Top 10, the original intention was to use these science cases to derive the scientific requirements for the VOb, and alongside the system architecture drawn up by the Project Manager at the time, to form the set of requirements specification for the system to be built.

Each of the cases described a topical astronomy challenge and contained details of the problems and potential VOb solutions. The science goal and the problem description were outlined, alongside the resources (e.g. datasets) required to solve the problem. In addition, the current approach to the problem was outlined, and a

contrast drawn with a solution using the VOb[36]. In some cases, the step-by-step approach of the VOb solution would be listed, along with a sequence diagram drawn up in Unified Modelling Language (UML) notation, a well-established formal language in requirements engineering, which captured the type of actions which the astronomer would have to perform in solving that astronomical challenge.

To the project team, writing use cases made sense: the project participants saw their role as supporting the existing programme of activities preplanned in astronomy by the research council, known as the PPARC Roadmap. Since future astronomy projects and funding would follow this plan, having the AstroGrid system supporting those scientific cases seemed a logical course of action. As the Lead Scientist states "... building a VOb system basically isn't going to set new scientific priorities over and above the ones that have been set by the peer community up to now" (Interview, AstroGrid Scientist, AG/5). This statement emphasised the project's role in supporting scientists work and underlined its goal in building infrastructure to that end. As a result, significant effort was placed towards the start of the project in devising, writing and agreeing the use cases.

For the developers, especially those who had little knowledge about astronomy, having the Top 10 helped them to gain insight into the types of activities and tasks that they were building for, and to grasp the type of broad scientific questions that the astronomers aimed to use AstroGrid to address. In this context, this offers some insight into how the developers needed to learn more than astronomical knowledge. They also needed to learn about astronomy as a discipline, including what resources astronomers required (e.g. datasets), and the existing practices of astronomers in using those resources.

---

[36] This comparison, between the existing and VOb approaches, would have no doubt emphasised the advantage of VOb use, as it would have highlighted the relative ease in solving these science challenges with VOb compared with other approaches at the time.

5.5.4. Shifting Techniques

As the project progressed, however, the developers required much more detailed information related to the use cases. While the use cases provided a good broad overview of the expected contexts of use, the team began to find limitations in using the same cases to derive more detailed requirements from them. It quickly became apparent that the effort required to turn these high-level use cases into detailed requirements was less straightforward:

> *" … it's certainly very different from when you design a spectrograph or a [counter?] or whatever scientific instrument. You do that kind of thing quite rigorously, you say, well, we want to be able to do this, so imagine if someone was doing this kindof observation and they want to analyse this and measure that then ok so we know they want that, so unless this camera had a resolution of this, and a wavelength coverage of this. And then to do that it has to have a lens of this size and you end up with a design. And it's based quite rigorously on - what are people going to do with this? and I think some software projects, can be like that …*
>
> *at first we went from use cases to activity diagrams and to classes and then and, well, when we did that the architecture … it was very hard to do that [for software] ... the kind of blow-by-blow use case style just really, didn't work well, I think" (Interview, AG P.I., AG/2)*

The breakdown from use cases to activity diagrams and classes ended up being too cumbersome as an approach for the team. Eventually, the Project Manager felt that it was too much work taking too long to turn science cases into technical cases, and the work was abandoned after the first few:

> *" ... we actually did do detailed analysis of a couple of them that did turn into real use cases technical use cases, going down to flow diagrams and things like that, but that proved fairly heavily going and too much to be sustained … when you're getting into very quick iterations that didn't get followed up. I'm still not sure whether that's a good idea or not, whether we should have kept trying to force that to happen but it didn't" (Interview, AG Project Manager, AG/4)*

After finding that the use case approach was not fit for their needs as the project progressed, the project team was expanded to include several 'Project Scientists'

who were astronomers working on the project, and could answer much more detailed questions about particular use cases. The hiring of more project scientists signalled a shift in the primary requirements elicitation technique; going from use cases to informal user interviews instead.

What is particularly interesting about this example is the way that a project team chose one particular requirements technique, and attempted to fit most of their requirements activities around it. Doing so worked to an extent. The Science Cases were useful as a way to communicate broad level requirements across the team, and also with astronomers and other interested parties outside of the project. But, as the project progressed the demands of the project shifted as the requirements needed to be much more fine-grained. In other words, the demands (or needs) on the requirements techniques changed. Viewed in this way, one can attach a sense of 'utility' to a technique. The utility can (and will) change as the e-science project evolves, especially since e-science projects have a longitudal element as mentioned previously.

5.6. Temporal Patterns in Development Work

Over the course of examining interview data detailing development work in AstroGrid, it became apparent that temporal patterns acted as important organisational mechanisms inside the development work. The concept was derived from Reddy and Dourish's (2002) study around information seeking in hospital work. Their focus was on how these rhythms inside work provided a resource for those engaged in that work for "seeking, providing, and managing information in the course of that work". (Reddy and Dourish, 2002, p344), Contesting the notion that information seeking is an individual action, Reddy and Dourish highlighted how information can be created between people, over the course of collaborative work. It

is through this information that those on the ward are then helped in interpreting and coordinating their search for data.

Here, the concept of temporal patterns is a particularly useful in analysing how project participants coordinated the requirements activities over the course of a long-term project. In addition, the process by which requirements were worked up, and recognised as requirements was also of interest. This is particularly relevant in light of an ambitious project such AstroGrid, where due to the high ambition and large scale aspect of the project, the number of components, scenarios and functionalities that had to be built and delivered in a coherent package seemed vast. Thus, there is a question of how it is possible to manage the volume and diverse types of requirements that are raised over the course of a project, i.e., how are requirements 'managed'?

Reddy and Dourish's notion of temporal rhythms and trajectories help to organise the multiple levels in which project participants coordinate their development, and generate information to help each other understand what requirements are appropriate to raise when (expanded on later). In this context, the use of temporal patterns point to the sometimes cyclic, but moving concerns that the developers have in meshing in different concerns, and prioritising particular types of requirements.

Coordination, in this context, stretches across multiple temporal and organisational layers because of the size and complexity of the project: from day-to-day work, to the general progression of the project, and to the broader environment in which this and other e-science projects are situated. While Reddy and Dourish's concept splits the cyclic nature of everyday work, with the linear trajectories as two core concepts, here a third is added to the analysis: phases.

The recognition of time and temporal aspects of development work has also been acknowledged elsewhere in e-science. Lawerence (2006) argues that there is a tension in e-science projects regarding getting the right "pace"; using the phrase to

hint at the difficulties in balancing, on one hand, the time needed to explore requirements sufficiently, and on the other, the need to move on with the project.

5.6.1. Rhythms in the Development Process

The Structure of the Project Development Cycle

The most obvious rhythm is the project's own development process which in the case of AstroGrid was split into cycles and iterations. Initially, the cycles were three months long, with shorter iterations that fitted into each cycle. By the end of the project, cycles were stretched to six months long, with weekly or fortnightly iterations in between.

At the start of each cycle, two meetings would be called with the development team: one was to discuss the scientific direction of the system, lead by the Lead Project scientist with participation from all the project scientists, the other was to discuss technical issues lead by the Technical Lead, alongside all the developers. Two reports were then produced: one detailing the scientific drivers, and one detailing the proposed technical work. It was not unusual for each cycle to have a theme to represent the focus for that period e.g., outreach, or scalability. These two reports were passed onto the Lead PI, who drew up the cycle's workplan from the reports (Interview, AstroGrid Technical Lead, AG/3; AstroGrid Scientist, AG/5).

Once the workplan had been agreed, the activities switched to 'implementation'. Each of the developers had their own area of responsibility, and they would form into ad hoc teams when it was deemed necessary for their work. The norm was that the developers were expected to implement the functionalities decided, and occasionally draw upon the project scientists for domain knowledge, such as understanding how an astronomer would approach a particular scientific problem, to provide feedback on a prototype, or for external contacts in the deployment of the AstroGrid infrastructure.

Rhythmic Nature of the Development Cycle

Because the development process was designed to be cyclic, the process fell into a natural rhythm. This pattern had several roles for the requirements activities in the project. The first point to note was that there was a process in the first place. This may sound a little obvious, but AstroGrid was the only project, out of the three case studies listed here, to have an explicit, formalised development process. Clearly, the philosophy of taking a "professional" approach had influenced the manager's method of structuring the development process. Thus, there was some structure to working out requirements, and it did not simply happen in an 'ad hoc' manner. However, for it to be effective, it relied on project members to follow the pattern.

Secondly, having this explicit process, meant that people would orient their understanding of which and what type of requirements were appropriate to raise at what time. Having this organisation mechanism provided project participants with an understanding of what level of granularity was being spoken about at those particular points. Requirements can be articulated at multiple levels. For instance, it could exist at a fine-grain level which only affects one component such as 'what colour does this button have to be?', or at a much broader level which could be the articulation of constraints on a system wide basis, e.g., 'what are the resource constraints in maintaining the system?' Dependent on the issue, numerous possible solutions exist, with a level of uncertainty in the feasibility and suitability of each one. This could lead to long discussions where no solution is settled upon. This is specifically challenging for system wide requirements, which participants note are important to discuss, but can get frustrating when it seems as if there has been little progression after long discussions. The phrase 'paralysis by analysis' is apt here. The struggle for all of the team members, but particularly for the project managers, was how to balance a sense of openness to continued discussions with recognising the need to move on, and the consequent close of a discussion.

The use of these rhythms then, provides information to team members to aid understanding of what level of granularity to speak at during discussions. For instance, the three-month team meetings provided a place to discuss wider, more system-level requirements that might affect significant aspects of the system. In contrast, on an individual level, smaller group meetings encourage team members to only discuss requirements that were specifically related to the particular component worked on at the time. On a finer-grain level, there might be individual decisions made between two team members. As a method to manage the overwhelming volume of discussions, questions and decisions, the AstroGrid team's attempts at enforcing a strict temporal cycle demonstrate the importance of temporality in the requirements process.

In practice, this resulted in project members being aware of the 'windows of opportunities' where certain types of requirements were more likely to be 'taken-up', or more likely to be "actionable" (Procter, Hartswood, Voss, Slack, Rouncefield, & Buscher, 2009). Specifically, suggestions for significant changes (e.g., a new function, a new tool or new implementations of components) are more likely to be incorporated into the workplan during the planning meetings, where all project members (scientists and developers) can make suggestions that would require significant work. Typical suggestions at this stage included datasets to be added, useful functionality or tools. For instance, in the extract below, one of the project scientists explains how she proposed a new tool (the Helioscope) at one of the planning meetings and it was approved:

> *"Interviewer: did you suggest the helioscope to them or did they suggest it?*
>
> *Scientist: no I suggested it actually ... well we started basically the whole thing was* ***most started at planning meetings*** *so [Lead Scientist] who is the project scientist and [Technical Lead] who is the Project Manager, were supportive of the idea and then, and they said yeah seems like a good thing to do and so I worked mostly with one developer from MSSL in London and so basically we worked on how we could do it and he did it [laughs] … it took probably three months" (Interview, AstroGrid Scientist, AG/9)*

Considering that the development of the tool took a non-trivial amount of time (3 months), it is possible to see that this development might have been disapproved if the proposition was raised at a meeting that was focused on more detailed requirements; i.e., the size of the request should meet the level of requirements being discussed at that point.

Conversely, once the planning meetings are over and the design and development begins, then more detailed requirements are likely to be 'actionable' at this point:

> *"... the requirements ... were often reasonably vague, so there was quite a scope for developers to interpret for themselves, and to suggest how things should be done themselves. We weren't given a very detailed spec and told to code this" (Interview, AG Developer, AG/10)*

For example, smaller, detailed requirements directly tied to a tool such as usability and interface would be more actionable at this stage.

Although it might not be completely conclusive that only major functionalities or tools are accepted at the planning meetings, it is not implausible to see how the planning meetings provided a forum for team members to review the overall progress of the project, and to offer suggestions on what to do next[37]. Once the planning was over, and the workplan had been decided, the developers, on the whole, were unlikely to deviate from that overall plan unless there was a sufficiently good reason to do so. They were given a vague description of the aspect that they had to implement, and it was their responsibility to investigate possible 'solutions'. Generally speaking, it is at this point that interface and usability requirements will be of concern to the developers, as they go through the tasks of working out more

[37] However, it is also important to note that these meetings included the whole team, meaning that a level of consensus had to be gained by the team for it to move forward. Although it is likely that there were intricate negotiations (potentially involving external political influence) that lead to particular tools or functions being included in the system, this is a point outside the scope of this thesis; but nevertheless, an important point to note.

detailed requirements, develop an understanding of the feasibility of possible solutions, and write the software. At times, they might even have to negotiate with existing organisations for resources in order to deploy their software. Most of the time, a developer would work with a project scientist, to use their understanding of astronomy in this process.

Any requirement may be raised at anytime during the project, but the argument here is that there will be certain 'windows of opportunities' where particular types of requirements are more likely to be incorporated into the workplan. To borrow a phrase from Voss et al. (2007), there is some level of "operational readiness" required, before suggestions can be taken on board. Although in their case, the authors were talking about the potential contributions of ethnography in systems design, their argument resonates with the discussion here:

> *"... it is also likely that in order to be receptive to ethnographic findings there has to be a sort of 'operational readiness' to take on-board the insights they have to offer – that certain sorts of finding become useful and actionable, and when the technical work has reached a particular stage in its maturation – where findings become opposite to the developers' technical concerns rather than competing with them. That is to say developers have to see not only the relevance of the findings, but also be in a position to do something about them." (p50; their emphasis)*

In the extract, the authors argue that the impacts of ethnographic findings contain limitations, especially when a developer has overriding technical concerns. A similar argument can be made here: scientific concerns could not be acted upon until the system had reached a certain level of maturity. However, it is not simply scientific concerns that are left aside until later on, but usability requirements as well.

Finally, the third implication of having a rhythm to the project was that these planning meetings provided an interesting role in project coordination, especially since the developers were physically distributed around the country. One important point to note was that despite developers being hired and managed by a partnering university, it was not uncommon for developers to work from home, where their location could be tens of miles away from their official work address. One of the

more interesting side observations to make about this arrangement was that the developers tended to be much less attached to the universities they worked in, but instead, identified much more with the AstroGrid project. One account that the Project Manager made was that since they had a distributed workforce, setting up and enforcing the adoption of communication tools (e.g., instant messaging, wikis) was one of his most important tasks, highlighting his priority of communication in a geographically-distributed project (Interview, AstroGrid Technical Lead, AG/3). Thus, despite the project using various forms of communications, project members rarely have a chance to meet each other physically. Such meetings therefore provide an important face-to-face opportunity for the project scientists to discuss potential requirements and express concerns.

Managing and Maintaining the Rhythm

Despite a codified, formalised development process that people were oriented to work to, it was not a straightforward, easy-to-manage process to maintain this rhythm. Instead, tremendous effort was required in maintaining and managing it; problematic issues included coordination across distributed sites and the length of the cycles.

Some developers questioned the effectiveness of the development process in coordinating across the different sites. Coordination across the sites was not simple, for instance, developers might have had different ideas of when a deadline was:

> *"... well in principle the project was running on 3 month iterations every single one was delayed and that was really, really poor ... what would happen is that some of us would finish on the three months and some of us would drag on for a few weeks, sometimes months and of course, the ones that have stopped kind of go, err, we can't do anything now because we've stopped and we're waiting for things to happen because there was all the problems to do with co-ordinating iterations ... iterative development is hard." (Interview, AG Developer, AG/8)*

As explained above, some developers would finish on time, whilst others might finish the bulk of the work on time, before taking a little more time to tidy. The developer above clearly attributes this tardiness to the challenge of coordinating multiple sites across geographical distances, and it is plausible that if all the developers were on-site, it might be easier to have shared rules regarding when to finish. Whilst this is clearly a contentious and complex issue for any Project Manager, the emphasis here is that such delays disrupt the rhythm of the development work.

Another factor which affected the decision/judgement regarding how long/short a cycle was, was the issue of how long to spend learning new technologies. On one hand, there was an acknowledgement that the team needed to spend some time exploring and understanding the limitations of the emerging technologies they were working with, and on the other hand, they had to make a decision and to move on. There is a related point discussed later on, where there is also a tension in the need to constantly evaluate and re-evaluate the future trajectories of the technologies the team was using. Lawerence (2006) also mentioned this issue of finding the "right pace" in balancing the time needed to explore requirements sufficiently, with the need to move on with the project.

Evaluation and Accountability Rhythm: Going from "Demo-to-Demo"

A second, on-going rhythm observed in the project, was one of evaluation and accountability. Unlike the previous rhythm, this was less explicit than the development cycle. There are two aspects to this: the aforementioned planning meetings were a way of evaluating progress internally, but the one that is explored here, is the regularity that the team had to account to the AGLI Board and the wider astronomy community.

Out of all of the project teams studied in this research, the AstroGrid team undoubtedly had the largest scrutiny from the community it sought to serve. This

pressure played a significant part in the prioritisation of their requirements throughout the early part of their project.

In practice, managing AstroGrid's engagement of the community was very demanding. It is fair to say that there was a certain level of scepticism aimed at the project from the wider astronomy community due to its ambitious aims. On the whole, the community seemed to acknowledge the issues that the VOb vision was attempting to address, i.e., the problem of the data deluge and the need for better data analysis tools, but was somewhat ambivalent since these issues were not of immediate concern to the 'average' astronomer. Working out how to deal with the data deluge from several large digital sky surveys was a long-term challenge for the entire community and was not a challenge that astronomers faced on a daily basis. In addition, team members felt that infrastructure building was 'invisible work' (Interview, AstroGrid PI, AG/2; AstroGrid Developer, AG/10) and until the majority of the components worked together, the system could not demonstrate its full functionalities. Coupled with the dramatic increase in scale and expense of the AstroGrid project compared to other previous astronomy software projects, the AstroGrid team felt the need to demonstrate the utility of the system, and justify the spending being allocated to it. In practical terms, this accountability was translated to the numerous demonstrations (demos) that the team were asked to give; in effect, they were held accountable for the funding allocated, and had to demonstrate to key stakeholders (within the IVOA and the wider astronomy community) that there was a sense of progress. The audience of the demonstrations would not only be limited to the astronomy community, but at super-computing conferences and the IVOA as well (Interview, AstroGrid Developer, AG/10).

The need to regularly perform demos became a somewhat disruptive influence on the general development cycle rhythm. At certain stages, demonstrating the system's functionality became the project's primary drive in prioritising its requirements during the bulk of the implementation phase of the project. As one developer said: 'we would do things depending on the next demo' and because the demonstrations

seem to be relatively frequent, it was not unusual for the project team to go from "demo-to-demo":

> *"initially at least, our requirements gathering was very much driven by which was the next demo we were going to be doing there would almost be a bit of a panic that we would need this feature by the next demo. And this makes it sound a little crappier than it was there was definitely a long term vision for where we should be. But the priority of in which order we should do things were definitely set by the demos" (Interview, AstroGrid Developer, AG/10)*

The above developer is quick to point out that this was not as erratic as it sounds; there was a long-term plan in place for the system.

The need to provide an account to the wider scientific community is not restricted to AstroGrid alone, but is also a characteristic of other large-scale, ambitious e-science projects (Lloyd & Simpson, 2005). It is fair to claim that the more ambitious and visionary the project, the more demonstrations and accounting behaviours were required. For instance, as is seen in the next chapter, the eMinerals project, which was labelled a pilot project, had a less demanding need to provide an account to outside stakeholders such as their research council. The expectations that come with attempting to deliver an ambitious project, presents a precarious situation. Providing demos can disrupt the rhythm of development work, as these demonstrations creates a time pressure which forces the developers to make a choice between writing code which would make the demonstrations work, and taking more time to write code which is more robust (Interview, AstroGrid Developer, AG/6; Interview, AstroGrid Developer, AG/8). At the same time, however, the need to hold the project team, and in particular, the project management team, accountable for justifying the time and the funding, is also an important part of ensuring that the money is "wisely spent" (Lloyd & Simpson, 2005).

5.6.2. Phases

Moving to a broader level, distinct phases of development work can be seen as the project progressed. Unlike the cyclic nature of the rhythms described previously, this temporal pattern is not iterative, but instead, linear. Here, a 'phase' is simply used to represent a period of time where there is a natural inclination towards a particular set of concerns during the development process. Different areas of concerns are brought to the fore and emphasis and priorities shift as the system reaches maturity. The purpose of this section is to highlight the project team's account of the phases that emerged throughout the project, and the shifts that occurred between these phases.

Officially, according to project documents, the project was funded in two phases[38]: AstroGrid 1 and 2. AstroGrid 1 was split into Phase A (feasibility) and then Phase B (implementation), and followed by AstroGrid 2 (deployment). However, from interviews with the development team, a much more complex and richer picture emerges where emphasis shifted from exploration, to technology driven, to user-driven. Inevitably, these natural phases will mirror those outlined officially, however, these phases stretched across the 'official' versions of the phases.

Exploration signalled the start of the project, and requirements tended to be expressed on the broad, system-wide level. Much of this phase was discussed earlier in relation to the Top 10 and the design of the generic architecture, where the requirements came from management and were driven top-down. On the science side, the emphasis was placed on drawing out the distinctiveness of the VOb approach. After the agreement on the science drivers and the generic infrastructures, the project shifted towards the bulk of design and implementation of the system,

[38] Which eventually became three phases.

which signalled the move to the second phase, where the project was predominantly technology driven.

This next phase saw heavy emphasis on the technical aspect of the project. Both at management level, and amongst the developers, there was the recognition that in order to "do anything useful", it was necessary to solve some fairly complex technical challenges (Interview, AG Developer, AG/6), for instance, federation of existing astronomical (large) datasets. The day-to-day management team were quick to point out that these technical challenges had overriding priorities at this point in the project:

> *"At the start of the project, I would have said it was achieving a technical solution [was the major challenge]" (Interview, AstroGrid Technical Lead, AG/3)*

Furthermore, the technical challenges were compounded by pressure from the oversight committees to produce "technically clever things". One of the project scientists noted this push and the resultant pressure overriding end-user concerns at this stage:

> *"I guess there are some stakeholders in early on, I think the fuss was this was a technological project that had to create generic infrastructure that might be useful for other people as well and that's because of the way of the funding and the review, the oversight came from things like, the grid steering committee so a lot of the push in the early stages was to show you were doing technically clever things [..] that it might mean you weren't doing anything useful to the end user ..." (Interview, AstroGrid Scientist, AG/5)*

It is fair to mention that this pressure created a friction against the goal of building dependable software, which is characterised by one AstroGrid developer as "something with a nice user interface with a lot of documentation"[39]. To an extent,

[39] See Working Software wiki page (url)

the tension between having novelty in the system, a trait often associated with prototype and research projects, with writing "dependable" software, was one that was never reconciled in the project.

While there was a pressure to produce "technically clever" software by the top management, (which might not be an unreasonable expectation in light of the level of funding involved), on the 'ground-level' there was a more basic struggle in getting to grips with some of the technologies expected to implement the system. Initially the plan was to use Grid components – after all, the project was named AstroGRID. However, within weeks of trying out the Grid middleware, the team was disappointed with its lack of usability and documentation. This seemed to be a common complaint across the Grid and e-science projects. There seemed to be a feeling that the merits of Grid technologies were over-sold:

> *"the difficulty with it was that the standards kept changing and the implementations were extremely unusable for want of a better phrase and so we didn't really get anywhere and originally AstroGrid was going to piggyback onto the back of OGSA and Globus, but err, because they were so slow and the documentation was so poor and it didn't really work very well as is the case in the new systems, we bypassed it and went to the web services instead - so we basically built our own grid based on the same principles but on web services" (Interview, AstroGrid Developer, AG/8)*

In addition, another issue noted by the development team was that they saw Grid technologies as a solution solving a different problem to the one they had. Their impression was that Grid middleware was better suited for problems whose solutions require large volumes of computational power, whereas in AstroGrid, they framed their 'problem' in terms of data management:

> *"Most of the e-science stuff, is letting people do things that were de-facto impossible in a reasonable length of time before – like enormous computations that you just have to have on the Grid. No one was ever going to get enough CPU power in one room to do it otherwise. That's not actually true in astronomy, because it's data complexity that's its limits rather than CPU limits in almost every case." (Interview, AstroGrid Developer, AG/11)*

As the above developer argues, the situation in astronomy is that data complexity is the limiting factor rather than the need for computational power.

In the end, the decision was made to move away from using Grid middleware to web services[40], and the team ended up writing most of the components from scratch. There was a strong sense that writing components was extra work that the project team did not expect to undertake:

> *"one of the tensions of building the project, the other tension, or challenge is the fact that the technologies that we're using is quite new. This whole idea of distributed computing is technologically its not a done deal, you can't just buy the components out of a Microsoft package and stick the things together its a bit more complicated than that so some of the underpinning technologies that we relied on especially in the initial instance when we set up the project weren't, if you like, fit for purpose as they might have been, so it meant that we ended up engineering components in which we hadn't initially been expecting to had to have engineered and we had to do those because this pressure to deliver some functionality in a certain timescale so therefore if there's no other solution" (Interview, AstroGrid Scientists, AG/5)*

Whilst, in retrospect, it might have been naive to expect that the Grid components would be 'ready to use' considering the novelty of the software involved, this was something that was not evident until the developers tried to work with the components available. This 'choice' between Grid and web services is a good illustration of the technical complexity that project members had to deal with.

Also, part of working out feasible 'solutions' to requirements involved a 'practical' understanding of the resources available to the team, as well as how their software might work in practice. For instance, around the UK existed several data centers, and they acted as repositories for astronomical data. Searching through these data repositories was essential to operations of the AstroGrid system.

There was a tacit assumption that the data centers would be a stable, reliable infrastructure for services to work on, and as a result, it did not occur to the developer's that this would not be the case. One developer, who talked about

---

[40] As detailed in Chapter 8

problems with developing the registry component (responsible for the list of available data sources):

> *"Developer: We took ages [in deployment] because we finally got hardware in most of the places that we wanted to publish stuff and I think it was, the latter half of the project, so that caused more difficulties as well*
>
> *Interviewer: Was the difficulty because of not having access to equipment?*
>
> *Developer: Basically yes, or having equipment that was casually available, so it would go up and come down and go up and come down*
>
> *Interviewer: How did that work?*
>
> *Developer: It didn't. It was horrendous. So we would have things like crucial breakable points like the registry where everything is always querying the registry to find out where the stuff is and who's who it seemed that would go down everyday or so: in the early days anyway, it settled down a bit later ...* ***so we would talk to somebody, say, at [data centre] who in theory was publishing data, but they just had a machine that was down half the time****, because somebody would ring up and go, 'I want this stuff', and they would say, 'oh, we'll get that going for you' and then they would run it and it [the machine] wasn't available all the time" (Interview, AstroGrid Developer, AG/8*

The developer talks about how break points in the registry could be traced back to one of the data centres having their machine being used for another purpose. Thus, the reliance of the infrastructure that AstroGrid was built on top of was tested. This examples serves to demonstrate that when building on top of an existing infrastructure, nothing can be taken for granted. That part of constructing a solution was not only limited to the software components inside AstroGrid, but the solution also required the consideration of the resources and reliability of the services that the system required to function.

Four years into the project (Summer 2005, during AstroGrid 2), the development team finally released the first version of the AstroGrid system. Upon its release, it was clear that a considerable level of pressure was lifted from the project team: this was expressed both explicitly by some of the team, as well as in the doubling in length of the project development cycles (from three to six months). In some ways, this release signalled a sense of legitimacy to the project, especially in the context of

the aforementioned scrutiny by the community. More importantly, from the team's perspective, the release symbolised the fact that the system had matured to a 'ready-enough' state to be deployed to the public. Their attention started to turn away from the technical aspects of the system to usability and end-user aspects of the system. This move marks the start of the third phase where the prioritised requirements were *scientifically-driven*. The phase was framed as making the system 'scientifically useful:

> *"the key challenge for AstroGrid now is to become scientifically useful and that means [getting] non AstroGrid related astronomers finding what we've done to be sufficiently useful that they wanted to hang around, that they needed and that's proven an interesting thing to try" (Interview, AstroGrid Technical Lead, AG/3)*

In this phase, the goal became demonstrating the utility of AstroGrid to the wider astronomy community. There were strong incentives to do so. As the project headed towards the end of their six year funding period from the UK E-science programme, the development team were acutely aware that any future funding would have to be from the astronomy community[41]. Thus, the viability of the project would depend on the team's ability to demonstrate the system's usefulness to the 'everyday' astronomer. In the context of science, the most explicit and measurable metric in demonstrating value is in terms of scientific papers. Astronomy is no exception. From the project scientists' perspective, the best way to demonstrate the value of the system was through the number of astronomy papers published as a result of AstroGrid software (Interview, AstroGrid Scientist, AG/7). In this situation, it is hardly surprising that the aim for the project scientists, at least, was in rallying more astronomers to write astronomy papers describing discoveries made using the AstroGrid system The hope was that other astronomers would notice, and in turn

[41] This was because the demonstration stream of the UK E-science programme was finishing, not because of the project's ability to get this funding.

seek out and try to use the system, hence driving up demand. However, whilst the aim is clear, with a measurable metric, how this could be done was a significantly more challenging question.

In practice, this meant engaging with a range of potential end-users: astronomers, computing officers and technicians. Their activities ranged from encouraging astronomers outwith the project to adopt the system, to working with computing officers in astronomy departments to help them install the AstroGrid client onto astronomers' desktop computers, through to negotiating with technicians or astronomers in data centres to give AstroGrid access to their dataset.

Interestingly, the team found that they had to be a lot more targeted in their engagement activities. For most of the project, community engagement activities were based on a "broad brush" model of community engagement: a 'high reach, low detail' tactic through giving talks at conferences or invited talks on the AstroGrid system. This method allowed exposure to a large number of astronomers at a time: namely from 50-100 people (Interview, AstroGrid Scientist, AG/5). However, as the team focused their efforts on user concerns, they began to pursue an alternative, more "high-intensity" strategy, where the project team ran two-day user workshops for external astronomers (cf. project scientists) in groups of 20-30. The goal of these workshops was to teach astronomers about how to use the system, to get them to think of a problem which they think could be solved using the VOb, and giving them help to carry out the task (Interview, AstroGrid Scientist, AG/5; AstroGrid Technical Lead, AG/3). This shift can partly be accounted for by the need to teach astronomers how to convert their research problems into logical steps to navigate around the AstroGrid interface. One common observation during the user workshops was how astronomers got frustrated after 'hunting' around the screen for the option that they were looking for; they consequently asked for help, with the typical exchange involving an articulation from the problem in astronomy terms by the astronomer, and then having that problem expressed by the project member in 'computational' terms (fieldnotes, AstroGrid User Workshop, 2006).

In addition to the focused strategy, the team also became more directed at particular types of astronomers. Younger post-doctoral researchers (PDRAs) and PhD students were perceived to be more receptive to adopting the system than more senior researchers, who might "have done astronomy for 30 years ... may not want to change to AstroGrid" (Interview, AstroGrid Developer, AG/11). But also, one astronomer turned developer[42] made the observation that traditionally, it is the PDRAs and PhD students that drove tool development due to the manual work they have to do:

> *"Traditionally, it's the PDRAs and the Grad students who actually drive the tools because it's intensively hands-on, and you have to sit in front of a screen all day in order to get anything significant done. Now, when we [the astronomy community] get the Virtual Observatory right, a lot of that manual work gets automated" (Interview, AstroGrid Developer, AG/11)*

Thus, there was a sense that if the team reached the younger astronomers, then the system would become more "embedded" in their work and the wider community (Interview, AstroGrid Scientist, AG/5). It was not clear which strategy worked best, although the team did shift from the broad talks to narrow workshops over the project. One guesses that any effective engagement strategy would require both strategies being employed.

In addition, team members had to negotiate for resources from data centres to make more datasets available within AstroGrid; another important strand in their community engagement. Getting buy-in was not only a lengthy process but one which involved understanding the amount of resources and capacity each data centre had. In practice, getting more data archives into the system often involved negotiating with data centres, and it is well acknowledged that they are heavily

[42] As mentioned previously, it was not usual to find a developer who had an astronomy background working in the project. He was part of the project at the start.

under-resourced (Interview, AstroGrid Developer, AG/6). Part of that required a certain level of negotiation with the people based at the data centres, and convincing them to give up some time and resources in order to get their data published on the system, together with the secondary issue that the AstroGrid system was rapidly evolving, meant signing on could involve an obligation for frequent work:

> *"what we found was that not only do the data owners not have time to do that kind of stuff and learn new stuff, particularly considering it was changing all the time, and also that very often they didn't have the resources that we expected them to have" (Interview, AstroGrid Developer, AG/8)*

The strategy that the project team took was to run deployment workshops in order to teach the data centre managers about putting their data online (Interview, AstroGrid Scientist, AG/5). The effectiveness of such a strategy is likely to take some time to emerge.

5.6.3. Trajectories: Anticipating Future Directions

In talking about temporal patterns in development work, it is important to recognise that the AstroGrid system was not built in isolation, but instead, was shaped and shaping the wider environment around it. Here, the idea is that one could imagine each major part of that environment as having its own trajectory, i.e., its own path that it is heading along. As the project progresses, so does the rest of the environment, some parts at a faster pace than the project and some slower. During the interviews, the AstroGrid team members talked about the need to anticipate future directions in the wider environment as it impacted on their assessment of the system's requirements, and feasible solutions. In addition, the developers also noted they needed to develop this awareness because certain decisions could have serious implications for sustaining the viability of the system. Whilst later in this thesis, this argument is elaborated upon by exploring how developers actively shape the trajectory of certain aspects of their environment, the aim in this chapter is a much

more modest one: to point out the different parts of the environment, or the ecology, that the development team had to be aware of.

Conceptualisation of project directions is a relatively established concept in requirements engineering. For instance, Krasner (1989) argues for the conceptualisation of requirements as dynamically evolving through the project's lifetime. He uses the metaphor of the requirements engineer as a skeet shooter: "[t]he expert skeet shooter anticipates the trajectory, or most likely path, of a target and shoots sufficiently ahead of that target's current position so that the shot meets the target at a future point" (p8). Bergman et al. (2002b), in their paper arguing for the need to account for the "political ecologies" that exist in requirements work, argue that requirements are dynamically constructed between stakeholders and their vested interest in particular outcomes. Whilst developers concentrate on negotiations and the conflict resolution aspect of building large systems with multiple stakeholders, the perspective here is a little less focused on the political aspects of requirements work. It is focused instead more on how developers have to be aware of the environment around them, and on the fact that their understanding of the requirements, and the possible solutions, are situated within the bigger picture.

As mentioned previously in this section, I want to outline some of the major parts of the wider environment which the members of the development team had identified as important to consider, in working up requirements and their solutions. Part of this section will be to describe whether these external trajectories are faster or slower than the trajectory of the project, plus how the external trajectories impact on the AstroGrid system. There seemed to be four 'trajectories' that emerged as important: the wider astronomy community, the IVOA, the (astronomical) data publishers, and the Grid and web services world. Note that whilst the word 'stakeholders' is usually used to describe individuals or groups that have a vested stake in the development of the system, and can affect the development of the system, here, it seems out of place because the scale spans further than one individual, or even groups of individuals.

Instead, I use the word 'trajectories' here to emphasise the temporal nature, as well as the significantly larger scale that these represent.

UK Astronomy Community

Obviously, the developers had to be aware of how the astronomy discipline evolved. The developers saw their role as building a system to support the future work of astronomers, and the development team had to develop a key understanding of how the astronomy world would evolve and what type of work the system would need to support in the future. As one can imagine, the rate at which the astronomy discipline and community evolves will be significantly slower than the rate of the AstroGrid project. Thus, through the project scientists, the project team were able to anticipate changes which could have had significant impacts on the viability and sustainability of the project with relative accuracy. However, there are instances where events in the astronomy community could cause the development team to re-evaluate whether what they had done was still relevant to astronomers. For instance, one of the AstroGrid Top 10 cases – the search for Brown Dwarfs – was 'solved' within the first few years of the project by some other means. The Technical Lead used this as an example of noting how the requirements were constantly changing for the AstroGrid system, and advocating their use of the iterative development process:

> *"...very often the requirements are changing underneath us. As an example, when I joined, one of the top ten science cases to be solved was a search for brown dwarfs. The theory was that perhaps brown dwarfs would make up a significant proportion of the dark matter in the universe and that if we could find these things then that would help ... but within about six months people had just about cracked the brown dwarf problem in other ways. But the techniques that we were using to crack the brown dwarf problem are generically applicable. So we carried on but we focused on a different science problem. If you don't release often; if we had just gone and away in a darkened room for three years and come back and said we've solved the brown dwarf problem they would have said, yeah, we've done that two years ago. So it [the iterative development process] keeps us current" (Interview, AstroGrid Technical Lead, AG/3)*

International Virtual Observatory Alliance (IVOA)

As mentioned previously, the IVOA is an alliance of all the Virtual Observatory projects. Since the IVOA played such a big part in AstroGrid (and vice versa), it is useful, at this point, to provide a bit of context about the organisation and why it was important for the AstroGrid team to actively participate in the IVOA forum.

Data sharing was an integral part of the AstroGrid system[43] and one of the major objectives is to allow users to search multiple data centres (which would often hold different data) in a seamless and transparent way. AG would provide a framework for UK astronomical data centres to publish data and services, allowing their data to be accessed in a standardised format. It became apparent at the start of the project that even if data standardisation and protocols could be achieved within the UK, this would be insufficient. The system needed to interrogate databases from international observatories in order to achieve the true vision of a Virtual Observatory. The need for global collaboration was apparent:

> *"[we have to] interoperate… with the sort of global resources so within our project, we've had to be active within the global standards organisations, to try to agree push along this standards process so that the sort of standards that you need to discover a data set or get a dataset are in place and agreed by everyone, so that our system can pick up American data for instance and vice versa, an American system can look at UK data so you're not sitting there in isolation and you're not trying to reinvent the wheel" (Interview, AstroGrid Scientist, AG/5)*

[43] Part of the need for VOb is that it is argued that discoveries are made in three ways; the opening of new parameters (e.g. X-ray and radio astronomy); finding connections between different observations of the same object; increasing the coverage of observations to search for rare objects (Djorgovski and Williams, 2005). Djorgovski and Williams (2005: 5) concluded that "this implies two kinds of discovery strategies: covering a large volume of the parameter space, with many sources, measurements, etc., as is done very well by massive sky surveys; and connecting as many different types of observations as possible … so that the potential for discovery increases as the number of connections, i.e., as the number of federated data sets, squared."

In order to achieve the vision of seamless data sharing, data standards had to be agreed on a global basis. Regular telephone conferences were conducted between the National Virtual Observatory (U.S.), Astrophysical Virtual Observatory (Europe) and AstroGRID (UK). International involvement by all three projects drove them to create an international body responsible for the coordination of VOb efforts. In 2002, the International Virtual Observatory Alliance (IVOA) was created between the above three plus a dozen small projects in smaller countries. The IVOA provided a forum for standardisation. Its mission is to "…facilitate the international coordination and collaboration necessary for the development and deployment of the tools, systems and organizational structures necessary to enable the international utilization of astronomical archives as an integrated and interoperating virtual observatory" (IVOA website). At the time of writing, a total of 16 organisations are part of the IVOA, with some more advanced than others. Each VOb project is a separate project with its own priorities, aims, and objectives:

> *" ... I think that, the virtual observatory is not one unified thing. It's an alliance of cooperating and sometimes competing groups. And we come together at these meetings - like the one in Canada to thrash out standards for doing various things." (Interview, AG Developer, AG/10)*

Whilst at first some basic standards were agreed in relatively quick time, as the project wore on, and the IVOA began to grow in size, it became apparent that the pace the IVOA was working at was much slower pace than that of the AstroGrid project (Interview, AstroGrid PI, AG/2). For instance, the Project Manager described this as a tension between "relying on standards development which is externally developed … [and] the time scale of which the standards have developed". In fact, team members were often vocal in expressing their frustrations in working in the framework of the IVOA; this was clearly attributed to the perceived slow pace that the IVOA works at. Whilst participants acknowledge that agreeing standards would take a lengthy period of time, they were also clearly aware that they had to progress with their own work.

Grid/Web Services/UK E-science programme

Lastly, compared to the project, the technologies that the team were using were evolving at a rapid rate, which meant that the project team had to constantly evaluate and re-evaluate their technology choices in light of major changes in that trajectory. Having the technologies move more quickly than the project meant that there was always the need to anticipate any changes to the technology landscape, especially those which might yield significant changes to the requirements of the project. In making judgements about the technologies to use for a particular purpose, the developer had to make some assessment of each technology's trajectory, e.g., by considering the following questions: what are its current set of functionalities? What are its future functionalities? How suited is it to the problem I am solving? How widely adopted is this technology currently and in the future? To an extent, difficulties can arise in instances where the implementation technologies become unsupported in the future, or where strong reasons begin to develop against using that particular technology. However, this task was not simple because the web services and Grid trajectories were moving so quickly. There is a certain sense of ambiguity and uncertainty associated in making long-term assessments. Here one advantage of adopting a component-based architecture was the possibility of using new components, in order to mitigate the risk associated with making these decisions during the early stages of the project.

One example is the AstroGrid workflow engine. In the early stages, the team decided to write their own workflow engine because they felt that there were no available engines at the time that suited their needs. However, the team were prepared to switch their approach (i.e. adopting another engine) should the external environment change so as to yield more advantages that way, e.g., if one workflow engine became dominant, this could be their workflow engine, or it could be some other engine.

The decision was being re-evaluated during the interviews, which took place 5 years into the project:

> *"But it maybe there's a clear winner now, in the way that e-science world in the UK is shaping up and now we're looking to implement a workflow engine from another community and we think that would be more robust" (Interview, AstroGrid Scientist, AG/5)*

In this case, they were considering switching to use the Taverna[44] engine from Biology. They perceived the shift would enable them to have a workflow engine which was more robust in the long term, as well as bestowing the advantage of having one fewer component to maintain:

> *"… so therefore if there's no other solution, which are [...] and that in turn has meant that hasn't had stopped still, stood still, so there are some areas some bits of the system which we're now having to engineer will replace with other components which has been won out in the wider computer science software world. So now the challenge is to integrate in components which are going to be more robust longer term." (Interview, AstroGrid Scientist, AG/5)*

The team had to be always mindful of the emerging landscape and aware that they might have to incorporate additional functionality, or refactor their solutions in light of changes. Thus, 'technical' choices are not final, one-time decisions. Instead, decisions are revisited over and over again. This could mean that their solutions have to allow for potential changes and to reflect their anticipation of future trajectories of the technological marketplace.

[44] http://taverna.sourceforge.net/

5.7. Discussion

To summarise, this chapter has detailed the first case study: the AstroGrid project. Even though the astronomy community had a history in working in large-scale distributed collaborations, they found developing the AstroGrid system a challenging endeavour. For a start, the system was to be crafted from software rather than hardware: a type of development that the astronomy community has not had much experience in. The project team had to work out a (software) development methodology for large-scale, distributed astronomy-orientated software as they were building it. While at the beginning, the project took a similar stance to the way previous astronomy projects had been structured – by examining the type of science needed to be carried out with the instrument and looking at the systems requirements for this - the project team soon found that the AstroGrid system required a different approach to previous projects, resulting in the team improvising their own software methodology.

One concept used in explaining the tensions and challenges in the development work in the project was the idea of temporal patterns. At the lowest level, were rhythms which were regular patterns that occurred throughout development work; 'windows of opportunities' were discussed, where certain types of requirements were more likely to be accounted for at particular times. Team members orientated towards this rhythm to develop an understanding of what type of requirements were most appropriate and when to bring them up.

A second rhythm of accountability was also described. Because the AstroGrid team was building an infrastructure for the entire astronomy community, they undoubtedly had the greatest pressure out of all three of the case studies in this research to demonstrate utility and value to the scientific community that they were serving. While the need to demonstrate usefulness was important, this rhythm was considered by the team members to be disruptive to their work; some saw this as a choice

between engineering software that made the demo work, and engineering software that was seen to be robust.

At a broader level, looking at the project as a whole, there were distinct phases where particular concerns were emphasised as the project progressed. Priorities at the start were on agreeing the science drivers and drawing up the generic component architecture that eventually dominated the rest of the project. The emphasis then shifts to one of technological drive, and the focus of activities was placed on understanding the technical complexities of implementing the system, and getting components to work together. Here the project team moved away from using the AstroGrid Top 10 for requirements purposes, and instead focused on recruiting a group of astronomers who the developers could engage with over the course of working out feasible 'solutions' to requirements. When version 1 of the system was released in mid-2005, the emphasis shifted to user concerns and making the system 'scientifically useful'. The science driven phase of the project saw extensive community engagement, which amongst other activities, meant that the team switched back to scientific concerns where they had to address issues of deployment.

Lastly, at the broadest level, was the need for the project as a whole to be aware (and anticipate) the paths of related areas, so called trajectories, because any major changes could have a significant impact upon the viability and sustainability of the project. This in turn impacted on the perceived requirements of the VOb as external trajectories shaped the project, and the project, in turn, shaped the external environment around it. I outlined three major trajectories in the wider environment: the UK Astronomy community, the IVOA, and the Grid and web services world. An important tension to point out here is how the project team have to manage the issue of working across multiple rates of progression. Some trajectories move faster than the project (e.g., Grid middleware), whilst others move more slowly (e.g., standardisation). The project team had to find ways to keep up with, or accommodate these different speeds. The important point here is that projects do not exist in

isolation, and having to manage across these different trajectories make working out requirements a much more demanding endeavour.

Overall, examining the requirements activities in the AstroGrid project has provided a glimpse into the complexity of building software to enable e-science. It is undeniable that the AstroGrid system was very ambitious, and will likely have brought its own unique set of challenges as a result. However, some of the themes outlined in this chapter continue to run throughout the rest of the data and discussion chapters.

Chapter 6: eMinerals

6.1. Introduction

The e-science design literature has often advocated lightweight, user engagement techniques, and in particular, prototyping. de la Flor et al. (2007) recommended, at the end of their preliminary research into design issues for e-science projects, that user engagement methods such as prototyping and iterative development should be used early on in the project to elicit requirements. Voss et al. (2007) argued that prototyping methods serve to bring users into the design process much more "strongly" and allow the designer to be more "responsive". Whilst being sound recommendations, they often remain simply recommendations, and do not explain how a prototyping or iterative process can be made to work in practice. This is especially of concern in the context of an e-science project, where the project teams are distributed, and thus could present a different scenario to more 'traditional' prototyping methods where the development team might have regular face-to-face interaction with the customer. The second case study in this research, eMinerals, provides the opportunity to examine the amount of work needed, including a careful choice of problem,in order to make the prototyping approach a successful one.

This chapter details the second of the case studies in this research. The project examined was *Environment from the Molecular Level* – shortened to *eMinerals* – a six year testbed to build a Grid infrastructure to enable larger, more complex

molecular simulations. They also had a broad aim of creating and working in a Virtual Organisation across five distributed sites in England.

In the eMinerals project, the team did not follow a formal software development process, but took an informal prototyping approach. Requirements were driven from the ground up by the scientists and emphasised the scientists' articulated needs as they started to use the Grid middleware tools. In practice, this played out through scientists trying existing or custom made prototypes/tools to do their work, and either finding them useful, or abandoning them if they did not. In this context, their approach to requirements work (and the development process in general) was similar to a rapid prototyping approach, where developers would produce a series of lightweight prototypes, then improve them through iterative user feedback. This case study provides a way to examine how some of the issues highlighted above play out in the context of an e-science project.

For the project team, team members noted the 'success' of the project in creating substantial changes in their work practices, hence encouraging a more collaborative environment than other molecular simulation projects, as well as some other significant changes. Some of the project members were explicit in advocating their "organic" approach (cf. model-driven development). However, rather than simply advocating prototypes serve as an effective method for requirements elicitation, this chapter takes a slightly different stance, by examining how setting up the environment enables the process to work effectively. By identifying and highlighting the efforts that team members made to drive the prototyping approach, observations from eMinerals can help examine the sorts of issues that have to be considered when planning and structuring an e-science testbed.

This chapter begins by providing some of the background and context to the science of molecular simulation, and the eMinerals project itself; in particular outlining the aims and tools produced by the project. It then illustrates, with two examples from the empirical work, the way that requirements emerged and the strategies employed by the team members to deal with them. This provides some context to the type of

environment which the eMinerals team worked within. The focus of the chapter then shifts to exploring some of the contributing factors to enable a prototyping approach to work in practice. In particular, the careful choice of a do-able problem and the commitment to an active user engagement process by the project team.

6.2. Background

6.2.1. Computational Simulations: Motivation and Practice

Molecular simulation scientists use computer simulations to model, predict and prove theories of how molecules react at the atomic level to test theories concerning their behaviour. Use of computer simulations has grown over the past few decades because they offer several key advantages over experimental science, namely in situations where it is expensive, difficult or even impossible to emulate under real-world conditions (Berman & Hey, 2004). For instance, in the context of chemical reactions, emulating reactions under a range of high atmospheric pressures is one example of an expensive, but relatively easy set of calculations that can be replicated using simulations. Relatively speaking, simulation science is a modern addition to the toolkit of the physical and life sciences. It is only since the recent advances in computation in the past half century that computer modelling and simulations have been able to flourish as scientific methods (Berman & Hey, 2003). As one can imagine, since their work is mostly computer based, it is fair to describe most computational scientists as reasonably computer literate, although this inevitably varies across the disciplines.

To simulate the molecular reactions, the simulation scientists draw from a variety of simulation codes; different codes offer distinct advantages over each other. Some might be fast, but provide only approximations, while others might take days, but offer high levels of accuracy. Accuracy could be achieved through different formulations built into the code. Scientists are likely to only be proficient in a few simulation codes, because it is unlikely that a scientist would need to learn to use a

new code for each new project. It is fair to say, that it is unlikely that a need will arise for scientists to know about all the simulation codes available. The scientists have to make a judgement about the problem they have at hand, the science they are trying to do, and therefore find a code which will enable them to conduct the simulation in an accurate way.

Unlike the astronomers in the last chapter, computational chemists tend to have less experience in large-scale collaborative efforts. Instead they are more accustomed to working in small-scale research groups, often on an individual basis (Dove, et al., 2005b). One reason to account for this is that simulation scientists are less pressured to raise funds for large specialised scientific instruments. It is likely that the scientists would be able to have access to High Performance Computing (HPC) facilities.

6.2.2. Using the Grid? The need for a Testbed Project

Due to the characteristics of their work, simulation scientists have a high demand for computational power (Dove, et al., 2005b). Scientists will typically have access to High Performance Computing or cluster computing facilities, although it was not until recently, that services such as the National Grid Service (NGS) were made available providing another computing facility that the molecular simulation community could draw upon. Whilst computing facilities are accessible to simulation scientists, the availability of computational power is expensive as well as precious. Access is sliced by time and it is typical for molecular scientists to have to book time on HPC facilities. Depending on the duration of the experiments, it is likely that the scientists would have to make judgements about how many and which simulations to run, in order to justify and validate their hypothesis. In this context the availability presents a bottleneck for the scientists.

With the e-science programme, Grid technologies offered the promise of significant, but unrealised, potential to revolutionise the access to computational power in simulation science. To the simulation scientists, the possibility of harnessing the

computational power of a pool of 'commodity' desktop machines could alleviate their bottleneck and have serious implications for the scale and type of scientific challenges that they could tackle. Since most universities have a supply of often idle machines for teaching purposes, being able to run Grid environments also proved an attractive proposition. The aim of the eMinerals project was to build a testbed to try out the suitability of using Grid technologies for molecular simulations.

6.3. The eMinerals Project

6.3.1. The Project

The eMinerals Project[45] was a testbed project aiming to use Grid technologies to account for as much of the environment as possible in molecular simulations. As a testbed project, its aims were to create large-scale simulations that could almost re-create realistic, real-world conditions (Dove et al., 2003). In addition, another aspect to the project is to create a collaborative infrastructure which will demonstrate how computational scientists could "accomplish the scientific objectives" (ibid).

Funded initially for three years (2001-2004), the project was awarded a further three years of continuation from the Natural Environment Research Council (NERC) e-science fund[46] (2004-2007). The eMinerals project was distributed over five Universities in England, with 12 Post-doctoral researchers and 8 Principal Investigators (PIs). In keeping with other academic projects, each institution had a Principal Investigator who managed the researchers on-site, although the PI's involvement in the project varied across the institutions. Overall, the PIs were responsible for the direction of the project. The number of researchers were spread

[45] The project's formal name is Environment from the Molecular level.

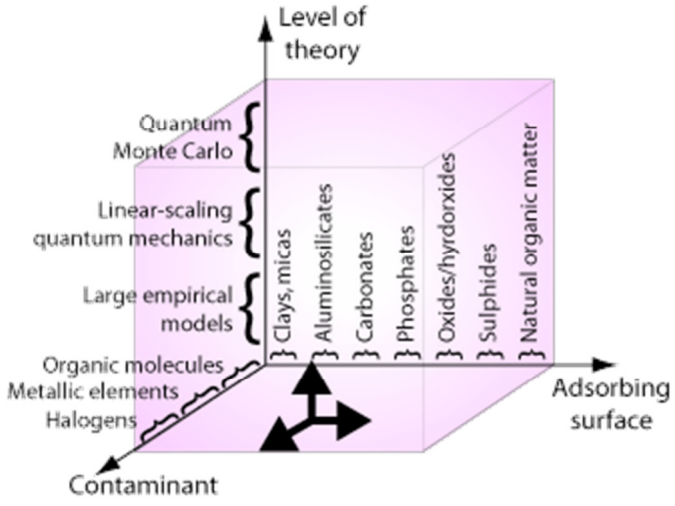
[46] NERC also funded two other projects – GODIVA and Climateprediction.net.

across the sites unevenly (again, not an uncommon practice), but the majority of the interviewees were based at the main site where the Project Manager, 3 PDRAs and one PhD student (who was not funded on the project, but fully participated in team activities) were involved.

The project had three broad strands to the project: *science*, *Grid tools* and *collaboration*.

In the science strand, initially the goal was to use Grid technologies to enable more realistic simulations by incorporating more aspects of the real world into the simulations. In simple terms, this aim required running bigger and more complex molecular simulations. The first phase of the project focused on two scientific Grand Challenges which tackled scientific problems that were "larger by orders-of-magnitude" than current simulations (eMinerals, eMinerals 1 Proposal, 2002), where these simulations would require novel work structure arrangements to take advantage of a more collaborative approach.

Figure 2: The eMinerals integrated problem approach (eMinerals, 2005)



In eMinerals 2, the focus was to take a much more integrated approach to the scientific problem that the project tackled as a whole. This move was an attempt to make a "qualitative difference" to working as a virtual organisation (VO) and act as a distributed "task-oriented VO that can pull together its personnel resources and

shared infrastructure to work as a collaborating team on large-scale studies of environmental processes" (eMinerals, 2005). Known as "the cube" the integral study aimed to "compare different contaminants and systems in a single integrated study", which is more ambitious than previous studies that have only focused on individual contaminants (Figure 2). In other words, the scientific drive in the second phase of eMinerals was specifically structured to tackle a larger, distributed study, pulling together the skills and expertise from the entire team in order to solve this problem. The Grid infrastructure and tools are described in much more detail in the Tools section in 6.3.2.

In the collaboration strand, the collaboration structure was referred to (in the literature, at least) as the eMinerals Virtual Organisation. The aim was to create an infrastructure that would support the communication channels to facilitate the type of collaborative science, as detailed in the science strand above.

The team was made up of three groups: *project scientists*, *simulation code developers* and *computer scientists*. The scientists' role was primarily to work on scientific problems, publish scientific papers, and to use the e-science tools that were being made available to them. Sometimes, the other team members would characterise them as "the customers" in the project (Interview, eMinerals Tool Developer, EM/9), and describe them as wanting "to be fed tools". The scientists themselves, acknowledged their obligation to contribute, and saw the other team members as producing tools that "make our lives easier" (Interview, eMinerals Scientist, EM/10).

The computer scientists on the project were based in the computer science departments at the partnering universities. Their role was, in part, to produce novel computational (Grid) tools to enable the science, to contribute to novel computational science research, to identify "limitations with current equipment" and to look at ways of enabling the scientists to do "whatever they want to do" (Interview, eMinerals Tool Developer, EM/4). Lastly, the simulation code developer's role was to develop extensions to existing simulation codes to make them compatible with the use of Grid middleware.

Although the divisions of labour between the groups seemed fairly rigid, in practice, these roles were less formalised, and the Project Manager was keen to allow team members to move between the groups if they wished to do so. For instance, one member was initially hired as a scientist, but ended up spending a substantial amount of time working on the mini-Grid and other Grid tools, although he had a lot of expertise in software development, having worked a number of years in industry development roles. Flexible arrangements are fairly common in academia, where researchers are given a level of autonomy in pursuit of their research interests. As a result, it sometimes leads to difficulty in accurately describing the group that team members belong to. Therefore, for the sake of consistency, I will refer to computer scientists and code developers simply as 'tool developers', because both groups worked with tools in the project.

Lastly, before leaving this section, one important aspect to draw attention to, is that the eMinerals project was very much seen as a testbed project, and therefore, did not have the same political pressure to produce a grand infrastructure or tools for the entire community, as was evident in AstroGrid. In eMinerals, the intention was simply to test out the Grid middleware tools in a molecular simulation context, whereas in astronomy, there was already the vision of the Virtual Observatory that heavily influenced the research council's (PPARC's) decision on investing in AstroGrid. To highlight the differences between the two projects, the eMinerals Project Manager joked during the interviews that:

> *"no, we didn't set out to do anything really [laughs] we set it out to try things out I guess, I think the expectation is of testbed projects is that they don't produce anything. I've heard that said on several different contexts actually. I think we have produced stuff, but that was never stated of us" (Interview, eMinerals Project Manager, EM/2).*

This was quite ironic, because it was apparent, that eMinerals produced some tools that, from the scientists' accounts, made an impact on their work as can be seen later on. The project team was aware of the need to address usability concerns and is

remarkably open in documenting their experiences as found in Dove et al (2005a) and Dove et al. (2005b).

6.3.2. The Tools

The eMinerals project used a variety of existing Grid middleware tools: namely Globus Toolkit, Condor, Storage Resource Broker (SRB) and Access Grid (eMinerals Website[47], 2009). One of the earliest tasks in the project was to build a mini-Grid: eight Globus and Condor pools connected across all the different sites in the project (Figure 3). It also had six SRB (data) vaults, across the infrastructure, managed by a central server. Essentially, the mini-Grid served a dual role not only being a computing resource that the scientists could use to run simulations, but also allowing the tool producers to test out various Grid middleware tools (including their own). Further details of the mini-Grid can be found in Calleja et al (2005) and (Tyer, Blanshard, Dam, Allan, Richards, & Dove, 2003).

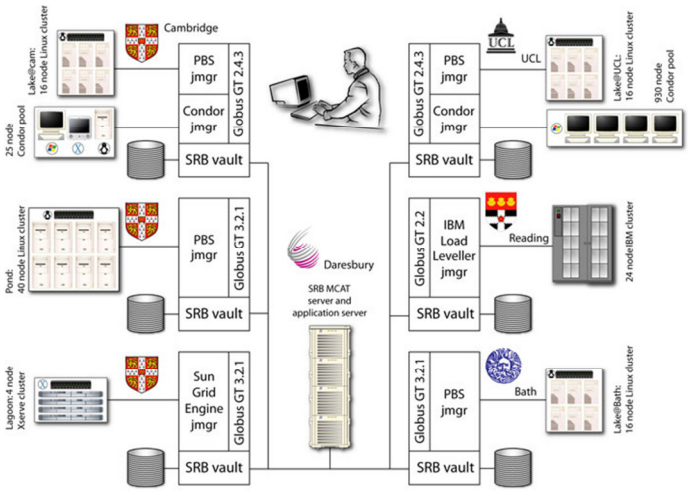


Figure 3: eMinerals Mini-Grid

[47] http://www.eminerals.org/tools/index.html

On top of the mini-Grid, the project team also produced a number of custom-made tools. These tools are split into two categories: science tools (to run simulations), and collaboration tools (to enable VO activities). In terms of the science tools, these often "wrapped" existing Grid tools to provide them with a better interface or abstractions to make these tools account for the specific needs of the molecular scientists. For instance, one of the tools which they produced was *TobysSRB* (White et al., 2006), the team's own SRB tool which wraps a new user interface onto the underlying SRB, citing existing "deficiencies" in pre-existing SRB interfaces as one motivation for producing the tool (ibid). Another example is My Condor Submit (MCS): a computer script that wraps around several middleware tools in order for scientists to run simulations in an integrated way (Tyer, et al., 2006). In terms of collaboration tools, the *Multicast Application Sharing Tool* (MAST) was built by two members of the team. MAST enabled application sharing (including PowerPoint presentations) and was deployed as a service in the Access Grid toolkit (Lewis, Hasan, Alexandrov, Dove, & Calleja, 2005). It was used as the platform to conduct the team's weekly meetings and was an integral application in the suite of collaborative and communication technologies used to enable the Virtual Organisation in the project.

6.4. The Prototyping Process

Recently, in requirements engineering and software development in general, there has been a movement towards the use of lightweight prototypes to stabilise requirements; essentially, the prototype would provide a shared reference for users and designers to discuss in future meeting(s). Feedback will be taken from the users and prototypes might be further improved. Prototypes can be anything from sketches on paper, to a 'rough and ready' version of the software satisfying a few major requirements. Some can be treated as throwaway prototypes, built solely for the purpose of eliciting new requirements (or agreeing existing ones) and are simply

discarded afterwards. Others tend to have more longevity, and are used as the early versions of full production level software (van Lamsweerde, 2009).

Prototypes, as an artefact for user engagement have become so effective that a variety of new software development methodologies, such as agile methods and extreme programming, advocate using an iterative prototype-based software development process. Generally speaking, the process starts by the users and designers meeting to talk over the users' requirements, and then a prototype will be produced from a subset of the initial requirements collected. An iterative prototyping process is often effective due to the closed-loop, intensive iterative user engagement (van Lamsweerde, 2009).

Specifically related to e-science, Ure et al. (2009) argue that the use of prototypes is an effective mechanism for the elicitation and alignment in requirements work for e-science. They identify a number of ways in which prototypes can be used as a "vehicle for engagement". In particular, they point to the way that early prototypes can prove valuable:

> *"The challenge of eliciting and aligning requirements for new and unfamiliar working practices can result in constant requirements 'churn' late in the process as users become aware of constraints or knock on effects on familiar working practices, or new opportunities that become clearer later in the design process. Early provision of 'something to hate' can act as a catalyst before requirements are set in stone. Engagement with the portal allowed users to be much more specific about what they liked and disliked to use it as a shared frame of reference." (Ure, et al., 2009a, p. 5)*

Using two e-Health prototype web portals to illustrate their point, they describe how early prototypes can serve to align socio-technical infrastructures and practices across the communities that they seek to support. By having a prototype, it is an "opportunity for grounded engagement with problems and possibilities in context", i.e., providing a shared reference point for users and designers when discussing potential problems and articulating future requirements. In summary, prototypes serve as a way to contextualise e-science technologies in practice, and allow users and designers to explore future practices through a physical artefact.

6.5. Making the Prototyping Process 'Work'

By the team members' own accounts, the use of prototypes seemed to be deemed a 'successful' and effective method of clarifying the emerging requirements in the eMinerals project. However, whilst the team found this to be an effective method of producing a suite of e-science tools, simply adopting a prototyping process alone should not be seen as the panacea in the development of e-science technologies. Instead, in examining the eMinerals project, a rich picture is built up of the subtle ways in which the process has been structured in order to facilitate the prototyping process; as well as the on-going efforts by the project members in their active commitment to working in an 'e-science way'.

As such, there are two types of factors that I have identified that relate to the issues outlined above. The first regards how various aspects of the project were structured to facilitate a prototyping process. The second relates to the active communication and collaboration that went on in order to effectively engage with the scientists. Understanding such efforts serves to point out the practical actions/mechanisms that have to be put in place in order to encourage this process to happen more effectively. However, before I start exploring the different factors, it would be relevant here to illustrate the prototyping process in eMinerals.

6.5.1. Background to Requirements Work in eMinerals

In eMinerals, requirements elicitation was carried out through the evaluation of lightweight prototypes, in an iterative manner. Similar to other academic projects, eMinerals took a more flexible, research-based approach, rather than carrying out an explicit, formalised requirements engineering process. Although the project team never explicitly follow a formal prototyping approach, there were a number of hints that they adopted a similar stance. For instance, one of the scientists said:

> *"... things will always change - and you discover new things. I suppose that this is the new idea in software development that you start with the simplest thing and you just improve it as you go on, and that seems to be a really good approach and I think that really worked for us" (Interview, eMinerals Scientist, EM/5)*

As one can see, the eMinerals approach to software development resonates with a rapid prototyping methodology, where a developer would speak to users about their requirements, prioritise them, and produce a rough, not-production-ready version of the software which meets most of the important requirements that the user specified.

Throughout the interviews were examples of how the scientists would use a middleware tool, find it useful and stick with it, or find it too difficult or not useful and move on. Project members often characterised the process as "organic" (Interview, eMinerals Scientist, EM/7; EM/2; eMinerals Developer, EM/9) implying that the process was exploratory: weaving scientific, social and technical elements together. Interestingly, team members advocated against having an explicit requirements engineering phase, instead arguing for a long-term engagement process, stating that a relatively flexible approach has allowed the team to explore and react to unanticipated requirements that emerged as the project progressed:

> *"I think it's enormously important that we do have the scientists and the tool producers working together reasonably closely on a team, in a dedicated project in a way that they can organically explore what can be done" (Interview, eMinerals Tool Developer, EM/9)*

In addition, the Project Manager talked about how "specific details ... are obviously adapting as time goes on" (Interview, eMinerals Project Manager, EM/2), and that building the various tools is "all organic, because you realise you need things bit by bit and they're driven by the needs – so you kind of didn't set out to do it; it just became apparent that we need this, and somebody did it".

In fact, as discussed later on, team members actively advocated against having an explicit requirements engineering phase at the start of the project. This can be partly explained by respondents having a strong sense of uncertainty and ambiguity towards the start of the e-science project, in common with all the other case studies. The

project team started with a set of Grid middleware tools that they could use and a set of scientific problems that they had to tackle (Interview, eMinerals Project Manager, EM/2). However, it was quite uncertain what e-science would 'look like' in practice. In fact, the Project Manager joked that they had a 'chaotic stage' (Interview, eMinerals Project Manager, EM/2) towards the beginning, describing the period of exploration that the team had to undergo in order to try out what was 'do-able', and very simply what was needed. This sense of uncertainty and ambiguity certainly resonates with the idea of the 'missing middle' described in the AstroGrid chapter, where there was a clear vision of what needed to be achieved, with some idea of the enabling technologies to realise that vision, but where it was much more difficult to understand how it would all work and play out in practice. By having a long-term prototyping process, the project team gained a better grasp of requirements as the project progressed by noting the way that the scientists reacted to the prototypes[48] produced.

So far, I have described the sense of uncertainty regarding what tools were needed towards the start of the project, and the fact that this uncertainty suited eMineral's prototyping approach. However, I want to ground these observations in empirical data to provide a flavour of the type and scale of emerging requirements that are being described. The first example concerns the usability of Grid middleware tools, and the second, data management tools.

Initially, one of the biggest challenges in running simulations in a Grid environment was the usability of the Grid middleware tools. As highlighted in Chapters 4 and 5, Grid middleware tools suffered from usability and maturity issues that were

[48] Interestingly, the prototypes produced were a hybrid between a 'throwaway' and the evolutionary prototypes outlined in Chapter 3. Small, proof-of-concept tools were produced, and they were discarded if the scientists did not use them, or did not ask for further development. In contrast, if the tool was found to be valuable, it would simply be continually developed (rather than thrown away). This is not particularly significant, but just merely interesting to note.

commonly highlighted by a number of different e-science projects. In eMinerals, the usability of the tools was described as a "show-stopper" issue (Interview, eMinerals Scientist, EM/7). As a result, the eMinerals team had to sink time and resources in re-writing or 'wrapping' the middleware tools for the scientists.

Getting the Grid tools to work, at this time, was actually reasonably challenging for two reasons. The first was linked with understanding the quirks of the middleware; for example, the scientists would be left confused as to why a particular job failed without notification. Using the middleware effectively requires an understanding of how the internal processes of the software work, so that the end-user develops a grasp of why the software fails (Interview, eMinerals Scientist, EM/7). The lack of error notifications, plus the lack of documentation, as well as, what were perceived as 'counter-intuitive' default behaviours (i.e., automatically deleting all the output files) meant that both scientists and tool producers found the middleware difficult to work with (Interview, eMinerals Tool Developer, EM/9). Another issue was to do with its interface (although the behaviour and interface are obviously interconnected). In particular, the issue was the lack of an 'intuitive' user interface for the simulation scientists, because it 'exposed' too much of the internals of the software[49]. The scientist often would not know the best option to choose and to an extent, did not need to, as default options were likely to work (Interview, eMinerals Tool Developer, EM/9). For the tool producers, the lack of usability quickly emerged as a problem that they had to deal with, because the project scientists were simply not using the middleware tools. To address this, they had to write an extra layer between the middleware and the end-user. Team members described this as "wrapping" the tools up that they had to work with (Interview, eMinerals Tool Developer, EM/9); providing an easier interface accounting for the specific user-requirements that the simulation scientists had:

[49] Including port numbers, locations and time outs.

> *"... part of the problem is that Globus and SRB ... they are also fairly low level implementations where they're not terribly useful. Globus on its own you would use for job submission its horrible its enormously verbose, it doesn't abstract nearly as much as it ought to from the end user you have specify far too many internal details. ... so we've ended up building, amongst other things that the additional layer on top that makes it more usable to non specialists, so that you don't have to specify all of the details of several locations and port number and time out and this that and the other" (Interview, eMinerals Tool Developer, EM/9)*

Their strategy to build abstractions on top of the existing technology was an interesting one, because doing so meant that they did not consider passing their requirements back to the middleware producers as a viable option. During one of the interviews, one of the tool producers mentioned the team's awareness that Globus and other middleware producers ran on a limited budget, and the chances of their requirements being acknowledged and incorporated into the next revision of the software would be minimal. Thus, building on top of the existing middleware was the best option at the time[50]. Once it was noticed that usability was a real issue that the team had to contend with, one of the team members wrote 'My Condor Submit' (MCS), which was considered one of the more successful tools built during the project. MCS was a computer script (which ran through a command line interface) that took a set of parameters from the scientist, read in the files from the distributed data storage system (called SRB[51]), submitted the jobs through the Grid environment (Globus) and then placed the returned files back in the data storage system. If the job failed, the script attempted to explain to the end-user why it had failed and retried the submission three times before the user had to take action. The MCS became a tool

[50] Pollock et al. (2007) for instance, mention the different strategies which groups of users would employ to get their requirements accepted in an Enterprise Planning System, where software has to be particular to each location for it to work, but yet, generic at the same time in order to allow the software to "travel" across different contexts of use. This also relates to a large discussion about e-science tool producers as consumers of middleware tools that de la Flor et al (2007) briefly address.

[51] The SRB was short for Storage Resource Broker – a Grid middleware tool that managed data files across a distributed system.

which some of the project scientists would use on a day-to-day basis. To emphasise the 'success' of this tool (and prototyping approach) it should be pointed out that this was one of the few tools that was used on a frequent basis (i.e., as part of the scientists' daily work) in the three case studies examined in this thesis.

This theme of unanticipated work carries through to the second example of data management. The Project Manager notes how eMinerals had always been seen as a computational project, but as it progressed, it became apparent that the data management side became increasingly important. He notes how "eMinerals was never seen by anyone as a data project, ok there's a whole load of other data projects, but it actually turns out that you can't really divorce data management from grid computing" (Interview, eMinerals Project Manager, EM/2). Extensible Markup Language (XML) was one of the key tools for data management at this time, and the team had found that one of the developers of the Chemical Markup Language (CML) – a specific chemistry based markup based on XML - was based at one of the sites. The Project Manager described how CML was "never on the agenda, but it became opportunistic" to develop some expertise in it. One of the team members was assigned to work with the creator, and together they started to incorporate CML into some of the other tools that the eMinerals team was writing, such as MCS. Seeking an outside collaborator to work with the project, rather than developing their own data standards from scratch was one strategy to leverage existing resources and practically manage the additional workload.

This example of adopting CML is interesting, because the astronomy community also had to produce a domain specific technology – Astronomy Data Query Language (ADQL), which was a version of the popular database language, SQL. Perhaps, these examples illustrate the potential need for domain specific components as a necessary part in the development of e-science applications and infrastructures.

However, to return to the argument regarding unanticipated – or rather – emergent requirements throughout e-science projects, from the MCS and data management examples. There was certainly strong evidence that both examples pointed to

substantial work that was unanticipated from the start of the project. In this context, having a long-term, prototyping process was suitable because it meant that the eMinerals team had the flexibility to adapt their workplans to fit around novel and emerging requirements from the close user engagement.

The eMinerals project, arguably, can be seen as 'successful' from the point of view of making a difference in the project scientists' work practices. As shown above, the project team, as a whole, created a number of tools which some of the project scientists used on an everyday basis, and the scientists' accounts provide a glimpse of the scale of change which is possible through e-science. However, using a prototyping process that had a long-term engagement was only part of the story. The other part, which is rarely told, is how these processes were made to work, and how team members had to actively make an effort towards this endeavour. In the next section (and for the rest of this chapter), the focus is on the factors seen to contribute to making a prototyping process work in an e-science project.

6.5.2. Two facets to Engineering the Process: Creating a 'Do-able' Problem and User Engagement

Earlier on, I distinguished between the prototype itself as an artefact, and prototyping as an activity, i.e, the process that the artefact is produced by. In examining the use of prototyping processes in the eMinerals project, it became evident that the prototypes were the artefacts that facilitated engagement between the scientists and the tool developers. However, upon deeper inspection, there were also a number of alignments made at the start of the project, which seemed to set up the 'environment' underpinning the development of prototypes that reflected the scientists' needs. These alignments were between the scientific problems to be tackled, the characteristics of the technology and the organisational structure (in the project, as well as the structure in the existing organisations, i.e., universities, they were working within). Making the alignments between these three aspects, and in turn, finding a suitable problem for the project scientists to tackle enabled them to use the

prototyping process to test out and provide feedback for the tools they were expected to use to accomplish their scientific goals.

However, creating the do-able problem was only one aspect of the overall process. Once the project was started, there were a number of activities which seemed to help make the prototyping process more effective; in particular, establishing a variety of communication channels, building rapport between the team members and learning to work in a multi-disciplinary team. Most of these activities have often been traditionally associated with user engagement, but it was also clear that because the team members were geographically distributed and multi-disciplinary, they had to actively work in making the engagement process work.

The focus in this section is to use the interview extracts to: firstly, point out how a suitable problem was found by aligning science, technology and organisational structure, then secondly, to show that user engagement activities are critical to fostering the effective development of effective e-science tools.

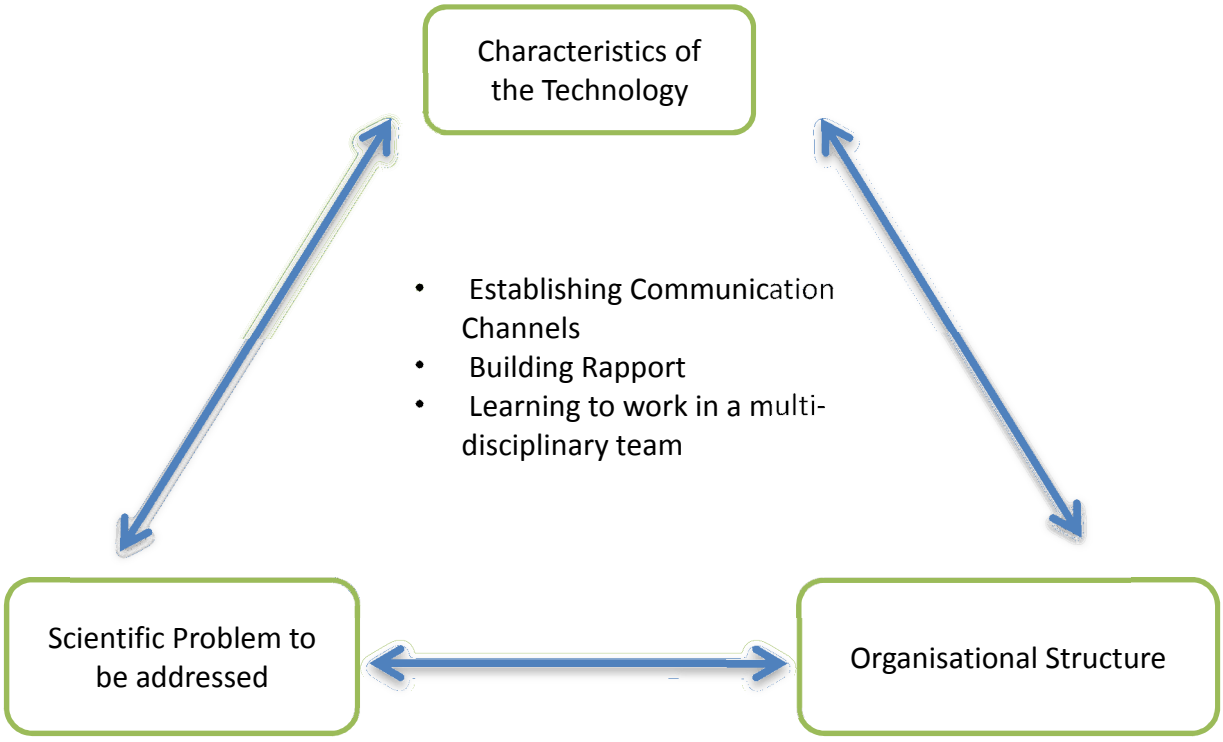


Figure 4: Making the Prototyping process work

6.5.3. Finding a Suitable Problem

In Sociology of Science literature, Fujimura (1987) argued that for a research problem to be considered as "do-able", the scientific problem had to be aligned on three levels: the experiment, the laboratory and the social world. Scientists have to conduct "articulation work" between these levels (i.e., coordination, planning and organising) in order to increase the do-ability of a problem. The emphasis here is on the notion that particular aspects of work have to be matched up in order for a problem to be considered as one that is feasible to tackle (and presumably fundable as well).

Interestingly, the notion that different aspects of work organisation have to overlap, in order for a problem to be considered 'do-able', resonates heavily here. From the data, one of the most interesting themes to emerge was one about the particularity of the scientific problem the eMinerals project chose to address because this problem

had to 'fit' according to certain alignments. By loosely borrowing this notion of multi-layer alignment, the discussion here attempts to point to three alignments that have to be considered in selecting the scientific problem to be tackled in developing e-science tools. But, rather than focusing on the elements themselves, it is the alignments between these aspects which are of interest here:

- Scientific problem and characteristics of the technology: Finding a 'Grid-able' Problem
- Scientific problem and organisational structure: Structuring the Scientific Work
- Characteristics of the technology and characteristics of the existing organisation: finding a Grid Friendly IT Infrastructure

These alignments are outlined in Figure 5.

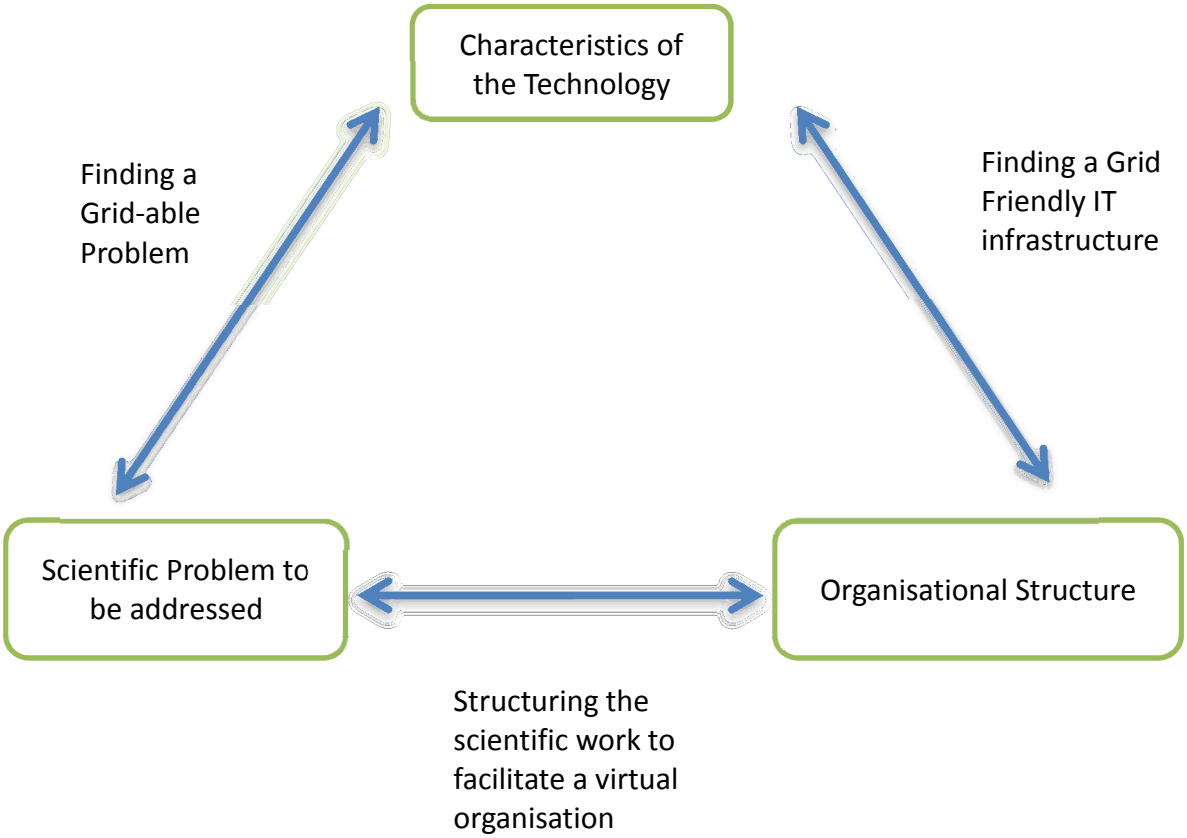


Figure 5: Finding a suitable problem

Aligning the Scientific Problem and Characteristics of the Technology: Finding a 'Grid-able' Problem

The first alignment in finding a suitable problem is in finding a 'grid-able' problem. A grid-able problem is a scientific problem that could match well with the characteristics of Grid technologies (or even e-science technologies in general). Given the context, the idea of a grid-able problem was quite surprising. Given that computational chemistry was, clearly, a computational based science, and that the scientists are arguably more computer literate than in other scientific disciplines, then initially at least, the expectation was that it would be relatively straightforward to

apply e-science tools and applications to their work. However, even in a computational discipline, not all scientific methods are amenable to Grid-type solutions. Instead, certain scientific problems have a particular 'shape' or characteristic that seems to be more amenable to e-science methodologies.

In the eMinerals project, the project team conducted a fairly specific type of work – known as ensemble work in their field - which compares the results from a number of similar jobs:

> *" well, a lot of our science is, I would guess some of our science is definitely high performance computing, but that doesn't fit into a Grid, but that doesn't fit in so easily with the Grid computing environment. I think if you're using HPC [High Performance Computing], then you can use some Grid-dy bits, by and large it's stand alone, you log in, you submit your job, it takes 12 hours and costs many tens of thousands of pounds to run but that's a specific example. So we have examples on that, but actually a lot of our work is, what I would call combinatorial and other people call it ensemble work; which is a slightly different take. Namely you have run a lot of copies of a very similar job. So you might for example run a job, run a study as a function of temperature and every temperature is a different job and those sorts of jobs you can fit onto a standard Grid component" (Interview, eMinerals Project Manager, EM/2)*

Here, the Project Manager explains that while some of the science problems in molecular simulation are more suited to a High Performance Computing environment, where the jobs are run on a single super-computer, ensemble or combinatorial work involves the submission of multiple simulation jobs on a similar range of values, which lends itself to a Grid environment because the simulation jobs can be run in parallel with each other. As a reminder from Chapter 2, the Grid middleware provides processing power through the orchestration of multiple computers, i.e., its advantage is from running multiple jobs in parallel. Thus, it is not difficult to see how ensemble work – a scientific method that relies on a series of small, similar simulation jobs – would be well suited to a Grid environment that enables multiple processes to be run in parallel.

As a quick aside, I want to show how well this alignment worked. From the project scientists' accounts, having access to this additional computational power has meant

that they could greatly improve the accuracy of their work because it could run more simulations:

> *'No it's [the e-science tools] a way of doing new things because well in a way, it's kind of both. But it's a way of doing new things because you couldn't have done it, it wouldn't have been possible to do it in the old way just because it's too time consuming, it's too difficult you know, you wouldn't you just wouldn't have thought about doing a whole load of calculations on 200 non-different congregates on a molecule you would have gone, 'ok that one is the most toxic, but this one is the one we've seen most in the environment, so we'll just take those two and do work on those and leave out the other 207. This is much better because that means we can do a sweep of all of them. And when you make assumptions like that, there's often something you're going to miss. Like with the box size calculations that I did, there was something that we missed because of an assumption but we found out because of what we were doing yeah, because of the way that we did it [...] you see broader trends and important aspects of the work that you've missed if you've had made assumptions and just pick different aspects of the [...] of the system' (Interview, eMinerals Scientist, EM/5)*

Here, the project scientist described how, previously, they had to be very selective in which parameters to choose in their set of simulation runs because they had a limited amount of time and resources. Since having access to the Grid environment, and middleware tools that help configure large volumes of similar jobs (which the team developed themselves), the scientist points to how, in the eMinerals project, they have the computational power and the configuration tools to be able to run 200 calculations in around the same time period. The scientist did not have to select or make judgements on the most important factors when running the simulations, but instead, could just run all of them, taking the "guess work" out (Interview, eMinerals Tool Developer, EM/9). Reducing the level of guessing was seen as a very positive step for the scientists. Another project team member, talked about how he ran a set of experiments with a particular set of assumptions in mind, and using those assumptions, selected which parameters to use to run the calculations with. Later on, once on the eMinerals project, he re-ran these calculations with a wider set of parameters, and found that his findings were incorrect because he had made some errors in his assumptions (Interview, eMinerals Tool Developer, EM/9).

While it would be foolhardy to assume that quantitative increases in the volume of calculations automatically translate to grand, qualitative transformations in scientific work, in this case, the scientists did convey the sense that having access to a Grid environment, coupled with tools that reflected their needs, allowed them to conduct work which they would have previously thought unfeasible due to constraints in time and resources. During one discussion, one participant talked about how it was possible to "parallelise the study" rather than simply the calculations:

> *"... traditionally people have parallelised the calculation - now we parallel the study. So we break up the study into different parts. One part may be a temperature, or a pressure or a molecule so we don't parallelise the jobs anymore, because the queues are powerful enough that you don't need to. But we basically, we run a lot of these points on the graph in parallel at the same time, all going off to different computers and the Grid methods allow you to do that very neatly" (Interview, eMinerals Project Manager, EM/2)*

In summary, this alignment actually suggests that particular types of scientific problems are more likely to lend themselves to Grid technologies, because the specific characteristic of the scientific problem aligns well with the advantages that Grid technologies offer. In this context, a scientific problem can be considered as 'grid-able'.

Aligning Scientific Work and Organisational Structure: Structuring the Scientific Work

In the second alignment, the scientific work in the project has to be structured in a specific way to help facilitate collaborative work between the project scientists, i.e., the scientific problem to be addressed has to be structured in such a way as to facilitate collaboration across the team, which is over and above the existing collaborative work. In order to effectively highlight the alignment, the contrast is drawn between previous collaborations in computational chemistry and the types of collaborations that eMinerals facilitated:

"...when I was doing my PhD I was involved in a EU funded collaboration ... which was 6 or 7 sites across the EU ... we met once a year and had a 2 day, 3 day discussion meeting. And the collaboration was effectively two people who work on a similar thing using different methods got together, or someone had a scientific problem which someone had some technology could solve some experimental or some computational [technique]. They effectively arranged to see each other's labs but other than that other than, this, everyone was working on the same big scientific problem when people could work together to solve a particular little scientific problem, they did, but there was no master plan." (Interview, eMinerals Scientist, EM/3)

"... the traditional way? well, everybody doing their own thing, I guess ok the traditional way of collaboration is to have an agreement to work together and then go back to your labs, do things and getting in contact after you've done something ... a traditional tool would [be] like email to keep in touch but by and large, people would have more or less a commitment to do something which they may or may not stick to, and then they go away and do things" (Interview, eMinerals Project Manager, EM/2)

As described by the scientists above, although there might have been large-scale projects in the past, it was rare for teams to work together in an integrated way in order to tackle large, complex scientific problems. The eMinerals team took the opportunity in their second phase of the project to experiment with working in a more integrated method, across the whole scientific team. Known as "the cube" (Figure 2), the structure combines three variables (different contaminants, adsorbing surface and different theories), and was structured in such a way that collectively, all the studies conducted by the scientists could be taken together to derive a more meaningful result than the individual studies could. As their 'All Hands Paper' (Dove, et al., 2005c) outlines, previously, the approach would have been to study individual contaminants on individual adsorption surfaces; however, with this deliberate organisation, the approach is much more integrated. In order to facilitate this, the individual scientists have to rely a lot more on working with other scientists in the team:

"...the stuff that I'm doing now we couldn't have got to the stage we've got to now without [Project Scientist]'s work in Bath, and he couldn't have got to that stage without our preliminary work here... That's a really good example of how the collaborations work because you know, [eMinerals Tool Developer] started off doing the calculations before I started here and he gave all of his results to [Project Scientist]. [Project Scientist] then used them in his calculations ... and then he gave us back our results that he got out, and now we're running them again with a different code, so its yeah, I mean really vital for

our operations that, because we don't have the expertise that [Project Scientist] has on site here, so you know, you need someone with his knowledge" (Interview, eMinerals Scientist, EM/5)

Essentially, this resulted in the overall scientific problem being organised in such a way that individual scientist's work builds upon the work of others, and data is passed around the scientific team. Results are passed from one scientist to another, and the above account concluded that this sharing enables expertise to be shared across the team.

The example here brings out two important discussion points. The first is that collaboration across scientific teams has to be, arguably, engineered. Providing communication tools is one step, but the scientists have to find a reason to share data and work together. Building in the need to collaborate presents one way of reinforcing this ethos. The second is that this collaboration was driven in a top-down fashion, with the structure being proposed by the eMinerals Principal Investigators from the outset, as an integral part of the e-science funding proposal. Although these two discussion points will no doubt have wider implications regarding the structuring and funding of scientific proposals in the future, here, in relation to requirements activities, the suggestion is much more modest. In relation to future e-science projects, this example points out the potential need to consider the division of labour during any future e-science tool development as a method of quickening the development of more collaborative tools.

Aligning Technology and Organisation: Finding a Grid Friendly IT Infrastructure

The third and final alignment is between the characteristic of the technology (i.e., Grid) and the IT infrastructure of the existing organisations that the project has to work within. In practical terms, what this meant was that there were certain universities that were structured in such a way as to lend themselves to running Grid

environments. In eMinerals, they found that one of the sites (University of London) was structured in such a way that it was possible for the pool of teaching computers to run as a Grid, as opposed to the Cambridge environment, where they had to build their Grid:

> *"in our plan? well a lot of these things are luck we had read about condor and people at UCL had talked to people and realised it was doable in UCL's context it's not doable in Cambridge context it depends on how universities organise themselves, and UCL had a very centralised way of handling teaching computers which Cambridge doesn't have and I don't know about other universities but UCL definitely had a very straight forward model we were prepared to use windows computers and the people running the computer service were very enthusiastic about working with us we found the place quite easily and actually all we had to do essentially was provide a computer that will act as a submit node and then learn how to use condor and the rest of it fell into place quite easily whereas there is a campus grid in Cambridge but there were too many segments of it that are behind different firewalls whereas UCL didn't actually have a firewall in those days so certainly there were firewalls between components on their campus grid so actually it was really quite easy to do" (Interview, eMinerals Project Manager, EM/2)*

Computing infrastructures often reflect the structure of the organisation itself, and the interesting implication here is that certain organisations will be more amenable to Grid environments than others.

In this section, I have shown three alignments between the science, technology and organisational characteristics in order to find a suitable problem that would facilitate an environment to produce effective e-science tools. In considering the discussion in this chapter about the prototyping process, it is important to bear in mind that the scientific problem had to be selected for the following three characteristics: (i) it had to be considered as 'grid-able', (ii) it had to be able to work in a prototyping environment, and (iii) it had to work on top of an organisation structured so as to be amenable to e-science technologies. These alignments provide the basis of an interesting discussion, which is not further explored in this research, but could yield interesting implications: are there particular aspects of scientific work in other disciplines which would be more amenable to the types of methods and principles that e-science advocates? And if so, how can these aspects be identified and exploited on a systematic basis?

Once the alignments have been identified and a do-able problem is chosen, with the project getting underway, another aspect important in making the prototyping process work, is user engagement, and active management of the engagement process.

6.5.4. User Engagement

Once a suitable problem is found, the other aspect to making the prototyping approach work is a commitment to active user engagement. In an e-science context where the project team is geographically distributed, finding ways to encourage users to provide feedback on the prototypes produced was expected to be a challenging prospect. Whilst the geography would be expected to cause constraints, it is apparent from the interviews that the participants seemed to have found a variety of ways to address this issue. In addition, another challenge often associated with working in e-science teams is the difficulty associated with working in a multi-disciplinary team. One developer highlighted the challenges of "hiding the complexity" in order to work smoothly across the team (Interview, eMinerals Tool Developer, EM/4).

In relation to the prototyping approach, one interesting point to emerge from the data is the appreciation built up by team members of each other's roles within the team. A substantial amount of active participation was required by project members to ensure that feedback was given by the project scientists, and the developers worked on tools reflecting the scientists' articulated needs. This commitment to the prototyping approach was achieved, in part at least, through team members appreciating the constraints their colleagues were under. For instance, whilst the project scientists are fully aware that they have an obligation to test out the e-science tools and technologies, the tool developers were aware that the scientists' priorities lay in conducting the experiments and finishing their work, rather than in submitting bug reports. They had to find ways to 'gently nudge' the scientists to provide feedback. Conversely, there were situations where it was also important for the project

scientists to develop an understanding of the Grid middleware and these are explored in-depth in this section.

Supporting a Variety of Communication Channels

In discussing the building of a collaborative environment, the most natural place to start is to look at the communication channels the project members had. One observation was the variety of communication channels which the team members had access to (Table 6).

Unsurprisingly, the use of emails and internet messaging were the most popular way for team members to pass on information on a daily basis, and both have been documented frequently in other e-science literature[52]. In addition, the eMinerals team would have weekly meetings via Access Grid, with team members taking notes on the project wiki during the meetings. The meetings served as weekly updates for the entire project team, and were relatively organised with a chair, and a pre-agreed agenda. Lasting around half an hour to an hour long, they served as a good way to catch up with progress across the project, and to discuss immediate future workplans (eMinerals Fieldnotes, 2006).

In addition, team members would occasionally travel to other project sites to give presentations or to help install software. Lastly, there were annual retreats and meetings such as the 'E-Science All Hands Meetings'. The retreats and meetings provided the whole project team with a chance to meet for a number of days to give talks, to review the project progress so far, and to discuss the tools that they had been

[52] For instance, Lawerence (2006) argues that one of the tensions in a distributed project is how much to communicate; on one hand, there is the need to include the relevant people in the email/conversation, but on the other, too much communication can be detrimental if it causes the participants to lose engagement.

using or building. However, while the retreats seemed to be run at the start of the project, it is unclear whether they were kept up throughout the project. Lastly, although this seemed to happen much less frequently, the Project Manager plus another team member would visit the project sites, and chat to team members about their work; the symbolic nature of these trips is discussed later on.

Establishing Rapport

Although communication mechanisms are important, they are only formalised, explicit parts of the collaboration process in the project between the team members. A much more difficult task is to build rapport across the team; sometimes, these activities are akin to 'team-building' activities. These activities are usually carried out in the hope of building an "effective team" which is essential for ensuring a successful outcome to cross-organisational projects (Lloyd & Simpson, 2005).

Frequency	**Mechanism/Channel**	**Communication mode**	**Audience**
Daily	Internet Messaging, Emails	Text-based	Individual, team mailing list
Weekly	Access Grid meeting, Wiki	Audio and Video based, Text	Whole Team
Occasional	Face-to-face meetings, Individual Access Grid meetings	Face-to-face, text and video based	Individual, Site-to-site
Annual/Infrequent	Retreat, All Hands Meeting, Visiting Trip	Face-to-face	Whole team

Table 6: Communication channels in eMinerals

An exploratory approach hinges on user feedback: prototype-based processes rely on requirements validation by users (Eberlein & Leite, 2002). In other words, the premise of a rapid prototyping process is based on having a small group of users for the developers to work with, followed by a period of close interaction with the users in evaluating the prototypes produced. The emphasis here is placed on the importance of feedback; agile methodologies and iterative development processes have always championed a user-based design. However, getting user feedback requires a level of *chasing* by the developers:

> *"...yes, the scientists on the project we get feedback from by the fact that they send the occasional email ... but, the problem is always extracting sufficient feedback actually. If something worked, [they] just ignore it and carry on and most people don't send off bug reports: 'so there's a bug it doesn't work, ignore it'. If they could find some other way to do it, then they will.* ***So there's a certain amount of pushing has to be done****" (Interview, eMinerals Tool Developer, EM/9)*

After building a prototype or a tool, the onus is on the developer to encourage the scientists to provide feedback, because whilst there is an understanding that the scientists should try the tools being released, the emphasis here is on *try*. Also, getting feedback is made more challenging in e-science projects for two reasons: geographical separation and incentives to do so. Firstly, the project team was distributed across different sites, meaning that there were few informal occasions for the developers to interact with the scientists, such as over lunch. Interestingly, this only became apparent when visiting the main university where three team members and one PhD student were based. Because the team were all based along a corridor, they would have lunch together, and it is these informal occasions that allow gentle 'nudging' to occur (eMinerals Fieldnotes, 2006). One tool developer observed that there was a certain amount of "peer pressure" that could be applied (Interview, eMinerals Tool Developer, EM/9), and one suspects that building and fostering a team spirit would make it easier for the developers to encourage the scientists to provide feedback.

Secondly, from the scientists' perspective, they have competing concerns. Like any scientist, they have their own priorities and workplan that they orient to, and unfortunately, contributions to scientific tool development are often not seen as so important as publishing scientific papers. This point is discussed in much more detail in Chapter 8, alongside examples from the other case studies, but for now, the point is to simply recognise that scientists were not reticent in providing feedback because they were lazy or difficult, but to acknowledge that they had work of their own. This tension is sometimes characterised in e-science projects as the tension between research versus development (see Lawrence, 2006; Welsh, et al., 2006).

Lastly, before leaving this section, there are two more observations to make about establishing rapport within the team. One is that the need for team building activities was more pressing towards the beginning of the project than at the end. There was a sense in eMinerals, that face-to-face activities (e.g. retreats, travelling to sites) were emphasised towards the beginning of the project, but as the project wore on, the need for face-to-face meetings diminished, as team members built up an understanding of the areas of expertise each project member had. An interview extract from one of the project scientists points to this. He noted that he would simply contact the "right person" directly if he had a query, rather than announcing it on the mailing list. This is slightly surprising, since researchers such as Lawerence (2006) have pointed to the tendency for e-research team members to discuss matters on the mailing list, in order to keep other members informed. This extract is particularly significant considering the interviews took place towards the end of the project, thus the team members had been given the chance to get to know each other. Lloyd and Simpson (2005) also strongly argue for the need for team members to understand each other's motivations, and this is done through openly encouraging the discussion of the motivations of team members in the early stages of the project. These discussions present a reasonably strong claim for the need to consider and schedule team building activities at the start of multi-disciplinary, multi-organisational projects to ensure the delivery of a 'successful' project.

The second, more powerful claim is that for this sort of rapport, team building needs to be driven from the top. In the next interview extract, one of the scientists was asked about what activities he conducted in gathering requirements, and he pointed out how he and the Project Manager would go to visit the rest of the team across the country and speak to them about their work:

> *"Developer: that people would prefer to talk more of a [...] especially when they give [...] or places just informal chatting, just about what their needs are so we've done that a couple of times sort of the whole lot once, then the whole lot twice, [Project Manager] and I*
>
> *Interviewer: did you do that once a year?*
>
> *Scientist: yeah, about once a year yeah, I think it was [Project Manager] who spotted the need for that or that would have been beneficial. [Project Manager] I think is quite good at getting the pulse of people. I wasn't convinced at first, but then when we did talk to people, yeah, this is a good medium between people because you don't see them, and they appreciate the fact that you've come down to see them ... you come [...] for lunch people loosen up and serendipity plays a part as well, people who've got a [...] which you just don't get from emails, because people just talk to the point and you don't sort of go on the beaten track" (Interview, eMinerals Scientist, EM/7)*

The scientist draws attention to the perceived need for the Project Manager to visit each of the sites, and the sense of appreciation of the team members for him to take the time out. It seemed that there was some sense that there might have been issues that team members might not be comfortable raising in a more 'formal' setting of Access Grid team meetings. Instead, having these visits meant that team members could air any concerns or grievances, and enabled them to meet face-to-face and build rapport. Presumably, having the Project Manager visiting team members would add more weight to the visits, because any discussion points were more likely to be taken up. The scientist in this extract was directly answering the question of the project team's requirements elicitation techniques; thus, there is a strong link between eliciting requirements across sites and the need for face-to-face, informal communication opportunities.

Working in a Multi-Disciplinary Team: Learning to Hide the Complexity

Over time, one of the most important skills for team members to learn was how to communicate and talk to each other within a multi-disciplinary team. In fact, the Project Manager noted how eMinerals was a bit of an "experiment" to see whether a multi-disciplinary team consisting of scientists, code developers and computer scientists would work. This learning process involved developing a 'good-enough' understanding of each other's work in order to convey the relevant information when working together. This problem was not confined to interactions between the project scientists and the tool developers, but was stretched across the team; after all, even the project scientists would have different areas of expertise. The problem did however remain most prominent between the scientists and computer scientists, as their areas of expertise rarely overlapped.

In relation to requirements work, understanding relevances was a skill that was necessary in facilitating the elicitation process. Scientists had to identify parts of their work that were relevant to the developers, which meant they had to learn 'enough' of the technical aspects of Grid infrastructures to be able to relate to the problems and issues raised by the computer scientists. One scientist, in particular, noted how much computing knowledge he had to learn: "at the beginning, I was really new to all this stuff. I didn't know, even Unix at the beginning ... all these e-science techniques scared me a lot, mainly because of the names, so I had to learn things ... and progressively I was learning more and more stuff" (Interview, eMinerals Scientist, EM/6).

Sometimes, it is difficult to manage this communications process. On one hand, there is a need for the scientists to understand the computing aspects because having a better knowledge of the types of technologies available will inevitably help the scientists spot opportunities for potential tools, on the other, there is a need to not overload them with technical knowledge. Interestingly enough, although the reverse can be true (i.e., not to overwhelm the tool developers with too much of the scientific details), the balance always leans towards the tool developers having to actively

manage the issue. This is because changes in the Grid middleware world moved at a faster pace than those in the simulation science world, especially during the early 2000s. Thus, while the scientists had to keep up with changes and constant updates to the various Globus Toolkit releases, the tool developers did not have the same demands on them to learn about new developments in the simulation science world. In the following interview extract a tool developer, who had been involved in the project for most of its duration, notes how the project scientists felt overwhelmed during the transition between Globus Toolkits 2, 3 and 4, and how a heated debate eventually broke out at one of the team's annual meetings:

> *"...we had had a long 2 day meeting about saying that, discussing all the new tools we had and all the new technologies that we would use, ... this was around the time that Globus 4 was being [...] and being used and so we got that one from Daresbury come down to talk to us the previous meeting oh yeah, this was actually my first project meeting, it was three, four months after I had arrived and apparently at the previous meeting he had come down and he described what Globus toolkit version 2 did and version 3 was in the offing and this would have wonderful brilliant new features and then he came along and started talking about Globus 4 and Globus 4 essentially threw away all that was fit into Globus 3 because they had a big fight, some people believe in web services, and other people didn't believe in web services, I don't know what, anyway he just came back and essentially, I mean, we were happy for him to talk, he was a very nice man but he was essentially saying yeah everything you were learning about Globus, you have to throw away, sorry, load of nonsense and said the last six months we completely changed our lives which got people's back up a little bit and it happened that a lot of the scientists were feeling a little bit over loaded because they had all these things fired at them which they didn't really understand" (Interview, eMinerals Tool Developer, EM/9)*

Thus, one of the most important skills in learning to work in a multi-disciplinary team was learning to discern what was *not* relevant, i.e., as one team member described, it was learning to "hide the complexity" of each other's work (Interview, eMinerals Tool Developer, EM/4). Talk loaded with technical jargon leads scientists to disengage with the prototyping process, as they become frustrated if they cannot make the conceptual link between how particular technologies may be relevant to their work. The skill lies in the judgement in identifying what is relevant and what is not.

As obvious as the above argument sounds, learning relevancy is still a difficult skill to achieve. From the interview data, this process required a certain level of perseverance. It is not uncommon, as exemplified by the above interview extract, that team members can get frustrated as they have to learn a lot, some of which might turn out to be irrelevant to the task in hand. As one tool developer notes *"...I think after a while, we sort of came to an overall understanding, we understood how to communicate with each other ... [now,] I can sort of guess what they need quite easily, and also we've already been working together so we all know what is available, everybody knows what each other wants, so, err, so it's not an issue anymore"* (Interview, eMinerals Tool Developer, EM/4). This has serious implications for future e-science projects as extra time and funding might be needed for new e-science teams to learn to work together; especially if team members have not worked together previously, echoing Lloyd and Simpson's (2005) argument.

6.6. Discussion

In examining the requirements activities in eMinerals, it is clear that adopting a lightweight, prototyping process for close user engagement is a valuable approach for producing effective e-science technologies that can create a 'qualitative' impact upon the scientists work. However, it is also evident that an effective prototyping process is not as simple as advocating the use of prototypes as a method of engagement, but rather, there is work and planning which goes into making the process a successful one. In particular, there are two aspects that have been highlighted over the course of this chapter: the need to find a suitable problem and the need to foster user engagement. Neither of these are easy tasks to accomplish.

In this chapter, one important observation is how project teams need to find a suitable problem. This is done through the alignment between the characteristics of the science, the technology and the organisational structures. Specifically, expertise

and skill is needed in selecting a 'grid-able' problem, and matching that problem with the organisational structures that the project has to work within. The interesting point here, is the sense that in eMinerals, the scientific problem not only had to be 'grid-able' but also one which would facilitate the use of a prototyping process as well. To a large extent, being able to find a scientific problem that can make these alignments is challenging, and it may be the case that finding a problem which fits all these alignments would prove much more difficult in other scientific disciplines. This is not to say that a prototyping approach would not work in other disciplines, but that certain disciplines (and more specifically, certain types of scientific problems) might be more amenable to the e-science paradigm than others.

Another important factor in making the prototyping process successful was the effort that went into fostering a collaborative atmosphere between the project scientists, and between the scientists and tool developers. The argument here is that in the context of an e-science testbed project, fostering a collaborative environment is a key aspect in enabling effective user engagement. Furthermore, the emphasis is placed on how this environment had to be fostered, i.e., encouraged, pointing out that this collaborative atmosphere was an accomplishment and provided a platform to encourage potential opportunities to be explored between the scientists and the tool producers. The project scientists were encourage to work together and to share their work and expertise because the scientific problem was structured in such a way as to integrate their work into the same problem (the cube; Figure 2). This discussion suggests that in the future, in order to achieve the change desired by e-science, part of the solution is to carefully consider the way that the scientific problem is structured.

The user engagement process was also discussed in length. A variety of communication channels were made available (Table 6), and there is a sense that team members had to be committed (and actively participate in) collaborative efforts over the course of the project. This aspect was an important part of the engagement process, because this communication was essential to overcoming the barrier of

working in a distributed team. An issue particularly important in the context of the eMinerals project, due to the need to find effective mechanisms to help engage the scientists in a dialogue with the tool developers across geographical distances. The findings seem to suggest that this is an active process and could benefit from a 'lead-by-example' approach from the management team. In addition, there is also some suggestion that in order to foster the collaborations, there needed to be a sign of commitment top-down i.e., from the Project Manager downwards. The fact that the eMinerals Project Manager travelled around the different sites to gather opinions and additional project requirements showed his commitment to the project.

A related point to the engagement process is that prototyping processes might benefit tremendously from having colocated teams. As mentioned previously, using a prototyping process almost needed a management of the user community. Having scientists and tool developers colocated may not only facilitate getting feedback for the prototypes through gentle 'nudging', but might also facilitate the communication process and learning to work together for a multi-disciplinary team. There might be some merit in considering different arrangements of personnel allocation for e-science testbed projects, with the suggestion of at least a core team based at one institution.

.

Chapter 7: TransProject

7.1. Introduction

Ambiguity of requirements in e-science projects have been noted by a number of recent papers. At a high-level, for instance, the difference between what can realistically be achieved in the project, and the promises from the funding proposal may be a source of ambiguity between the stakeholders regarding the expectations of what the project will deliver (Warr, de la Flor, Jirotka, & Lloyd, 2007). Consequently, the authors note how the Project Manager (of their case study, eDiaMoND) said that the process of "moving from the proposal to the project was an art" (ibid, p3).

Understanding requirements for e-science technologies is often characterised as being challenging by project participants throughout the fieldwork, especially at the start of the projects. From the developer's point of view, turning the project on paper (proposal) into reality (working system) is about moving from high level requirements, to a lower, more detailed 'nitty gritty' version of the requirements. This process generates a multitude of questions arising towards the start of the project: *What components are needed?, What technologies should be used?, Should that technology be built or bought in?, What resources will the partnering organisations contribute to the project?, What would the infrastructure be? and What are the future working practices that need to be supported?* It is the contention in this chapter that part of the ambiguity (and uncertainty) arises because of the

multitude of choices available, and that project members have to develop a practical understanding of the feasibility of particular solutions/choices. In e-science, this situation is compounded because of the number of options available due to the multiple partnering organisations, and the amount of learning that has to be done by each project member because of the multi-disciplinary nature of the project team (learning both in terms of domain knowledge, and the political/social arrangements of the partnering organisations). Thus, it is challenging to grasp because it is often difficult to see how all the parts: social, technical (components), organisational (working practices) and scientific (types of work to be supported) will mesh together into a feasible, working system which will support the types of work that e-science envisages.

This chapter takes a more detailed look at the everyday, practical work of developers, and the strategies they use in stabilising requirements over the course of the development process. Unlike the previous two case studies, where the data was primarily interview based, in the TransProject case study, a ten-month observation-based study was undertaken[53] that provided an opportunity to examine the sorts of uncertainties mentioned in the other projects in detail. This complements the discussions of the temporal patterns of an entire project as seen in AstroGrid, and the prototyping, user-engagement discussions in the eMinerals case study.

TransProject was a six year data infrastructure project in the discipline of translational cancer research. Its aim was to assemble a cohort (i.e. collect a dataset)

[53] Data was primarily collected observationally over the course of 11 months during the first and second years of the project (Jan – Nov 2005). This included numerous informal conversations with project members about their work, such as their approach, design and perceived challenges. This was supplemented by two short semi-structured interviews with project members, alongside project documentation. The project's weekly Tuesday morning meetings also provided a rich setting to observe since negotiations about the direction within the project were discussed. In addition, a considerable amount of observational work occurred, with an average of 2 or 3 days per week based at the office. In addition, a number of meetings with a related project – CancerStudy – were observed and noted as well.

of cancer patients in order to support a range of research activities from early-stage bench science (e.g., epidemiological studies) to clinical trials.

The chapter is structured as follows: firstly, some background and contextual information is provided: a background of the translational cancer vision is outlined; this is followed by a more detailed look at the motivations behind data re-use and electronic (record) linkages between organisations for translational cancer research; then background on the project itself; and finally, a brief description of the system to be built by the project team. This is followed by a brief overview of the literature on the practicalities of development work, alongside an exploration of why requirements work is challenging in an e-science context. Using two examples from the fieldwork, the section illustrates why it is often difficult to find feasible solutions due to the complexity and novelty of the situations that e-science technologies have to operate under. The attention is then shifted to stabilisation i.e., the process of getting a working agreement in order to progress and implement (at least, part of) the 'solution'. Five different techniques were identified from the empirical data in stabilising requirements, and these are described, alongside empirical data to support them. Then the chapter finishes with a discussion on a review of the themes explored in the chapter.

7.2. Background

7.2.1. The Translational Cancer Vision

The translational research vision aims to quicken the process between basic bench science and the delivery of clinical trials (Wehling, 2006; Webb & Pass, 2004), leading to the faster delivery of bedside patient healthcare. For decades, medical research has been perceived as a split between biological research in the laboratory, and the delivery of bedside medicine (Cambriosio et al, 2006); translational science has arisen because of the drive to close the traditional division of labour between these two seemingly distinct areas of knowledge:

> *"The necessity to develop this discipline reflects the schism which has been brought about by the separation of medical teaching and pharmaceutical research into "preclinical" and clinical issues. Bridging this chasm is crucial to success, if we want to use laboratory data to finally cure diseases in humans" (Wehling, 2006, p. 91)*

In certain cases it has been reported that the gap could be up to 20 years (Brekke & Palinkas, 2007), although there are encouraging signs that this gap is closing for certain areas of medical research (Cambrosio, Keating, Mercier, Lewison, & Mogoutov, 2006). In addition, there are signs that the problem that translational science addresses is being more widely recognised. For instance, in the United States, one 2004 US Presidential Candidate proposed the funding of a fifteen billion US Dollar programme for Translational Cancer (Pardridge, 2003).

Translational research focuses around collaborative work across multiple disciplines and across multiple organisations. Celis and Gromov (2003) outline the need for different medical professionals as well as industrial partners in order to collaborate, and highlights similar structural changes to the cancer research process in order to facilitate translational research:

> *"The implementation of discovery-driven translational research will not only require coordination of basic research activities, facilities, and infrastructures, but also the creation of an integrated and multidisciplinary environment with the participation of a dedicated team of clinicians, oncologists, pathologists, and epidemiologists, as well as industrial partners. Issues related to sample collection, handling, and storage, number of patients, availability of normal controls, tissue banks, quality of the clinical information, follow-up studies, and ethical considerations are critical, and must be carefully considered. As we redefine the manner in which we approach cancer, changes also need to be made in the way cancer research is funded." (Celis & Gromov, 2003, p. 9)*

The Translational science vision clearly resonates with the multi-disciplinary and collaborative nature of e-science.

Translational research, although gaining in popularity during the past decade, is still relatively young. Similarly to e-science, translational science is not restricted to a single area of research. Instead, the vision of translational research can be transposed

in a number of different contexts. Of particular interest in this chapter is *translational cancer research.*

Translational cancer research is, as the name suggests, the application of translational science principles to cancer research. In 2002, the UK Government funded the National Translational Cancer Research Network (NTRAC) as part of the National Health Service's (NHS's) commitment to translational cancer research[54]. NTRAC consisted of 15 centres across the UK all with the aim of facilitating different aspects of translational cancer. Later on, in 2007, the work was continued through a second funding programme termed Experimental Cancer Medicine Centres (ECMC), with a total funding of £35 million. Roughly speaking, each chosen centre was allocated £2 million worth of funding for anti-cancer science. The case study in this chapter is based at one of the NTRAC funded centres, which got additional funding from ECMC. The next section outlines the motivation for the research behind the work conducted at this centre.

7.2.2. Data Re-using and Creating Linkages: Motivation for TransProject

Currently, in order to carry out an epidemiological study, a number of complex steps are involved. In simplified terms, the process starts by obtaining ethics approval from the local regional ethics board on the design of the study. Every study that collects or stores patient data has to gain ethics approval before approaching patients. The application process is sometimes seen as lengthy, since the application has to contain detailed Standard Operating Procedures (SOPs) for the patient data obtained, what it will be used for, how it will be stored both during the study and after the study and most importantly, the purpose of the study. If the cohort will all be recruited from the same NHS regional jurisdiction, then only local ethics approval is needed; a different

[54] http://info.cancerresearchuk.org/news/archive/pressreleases/2006/october/230014

type of ethics approval will be needed if more than one NHS health board is involved.

Once ethics approval has been granted, patient recruitment and data collection can begin, including the recruitment of patients of interest. In order to recruit patients into the study, a set of ethical questions regarding how to approach the patients has to be answered. Patients can only be approached at the correct time and it must be ensured that they know exactly what they have consented to. Afterwards, additional data such as tissue samples may be collected. In the case of tissue samples, care must be taken to ensure that they are prepared, analysed and stored in a consistent manner.

Another aspect is that there might be certain sorts of medical records included in the cohort, e.g., cancer registry records, mortality records etc. Existing methods rely on transfer through paper or floppy disks. The process normally involves the researcher making a manual request through a form, and if approved by the data-releasing organisation, the data will be transferred on paper or on disk to the requesting organisation. The entry into the project's database is normally by manual re-entry. Only at that point can the epidemiology study be carried out.

There are two aspects to this research process that provided an attractive problem for e-science. The first is the potential to re-use data: *collecting patient data is expensive*. Building up a cohort can be time and resource intensive, and at the end of the study, under present practices this data would be destroyed so that other studies could not re-use it. By having a centralised dataset, there would be little need for future studies to build their own cohort. Of course, one of the most attractive advantages of having an available cohort is that this would shorten the research process for future studies by having the dataset readily available to carry out research with, rather than having to go through the whole process of building one from scratch.

Another aspect that could quicken the existing process is to develop electronic links to replace the manual printout and re-entry of the medical records as listed above. Developing a computing infrastructure that could facilitate sharing between different

medical organisations, including record linking between the repository genomics and proteomics laboratories, plus other healthcare organisations (e.g. National Cancer Registries, NHS) could provide great benefits. Potentially, it would collate important information about the current cancer episode or previous episodes (if applicable), it might provide information on treatment regimes or notify the repository if a patient died. Thus, more up-to-date and more detailed records could be kept about each patient. However, such data collection could only occur through the agreement of both the patients and the organisation involved. Due to the nature of the records held and the Data Protection Act (DPA), which prevents misuse of personal data, getting such organisations to release data would be one of the challenges that the project would encounter.

7.3. A Platform for Translational Cancer Research: TransProject

7.3.1. The Project

TransProject was a six-year project to build an infrastructure to facilitate translational cancer research, starting in 2004, with first phase funding ending in 2007, followed by a second phase of funding from 2007 to 2009. TransProject was a joint collaboration between seven partners, including the National Health Service (NHS), the National Cancer Research Initiative, as well as the local university's Public Health, Genomics and Computer Science departments. With seven full time staff and ten Principal Investigators, the project is the smallest out of all the case studies in this research.

The aim of TransProject was to build an IT infrastructure that facilitates translational cancer research, focusing initially on Breast and Colorectal cancers. Unlike other e-science cancer projects such eDiaMoND, a significant part of the project was to generate the cohort itself (i.e., collect patient data), rather than data federation. Broadly speaking, the aim was to accrue a cohort (i.e., produce a data resource) that

could be re-used over and over again for different cancer studies. Eventually, the data platform would support the translational research process on two levels: (i) it would provide support for multiple types of cancers, and (ii), it would support different stages of the research process (from collecting patient data for epidemiological purposes to facilitating clinical trials). TransProject was often seen as a follow-on project to a similar study (called CancerStudy here) that also collated data from colorectal cancer patients. By the time CancerStudy was coming to the end of its research, the project team had built up substantial expertise in data collection. For instance, they developed a set of Standard Operating Procedures (SOPs) and had built up relationships with partnering medical organisations (including the National Cancer Registry). As a result, the TransProject team would often call on the knowledge of the CancerStudy team, when discussing the issues and challenges in understanding the domain (both in cancer research and the practical work of the hospital wards).

The project would create a central repository of patient data that oncologists could use for their research. A mixture of data collection methods would be available, including: a) data which was collected by the project team, specifically to be stored in the repository and b) data from other healthcare organisations (e.g., medical records from the NHS or oncology history records from cancer repositories) regarding those patients. Part of the proposal was to use Grid technologies to enable the transfer of data from other organisations, to enable the storage of large volumes of data, and to facilitate data linkages between multiple sites and laboratories.

The project had seven full time staff: a Project Manager, two field research nurses, a research technician, two developers (e-scientists[55]) and a database curator/manager.

[55] One point to note is that although they were officially named e-scientists, they had the same role and responsibility as a software developer, thus for consistency, they are referred to throughout this chapter as developers.

Broadly speaking, the responsibilities of the team members were: the research nurses would collect patient data, the developers were expected to develop the IT infrastructure and the database curator would ensure that the quality assurance of the data entered into the system would be high. Interestingly, the team, whilst being very multi-disciplinary, did not have any (embedded) project scientists like the other e-science projects in this research. This proved to be rather important, since there was no clinician working on a daily basis with the project[56] (cf. AstroGrid and eMinerals).

Compared to the other case studies in this research, there are three characteristics that made this project distinct from AstroGrid and eMinerals. Firstly, despite having a similar character to other e-science projects, TransProject was not funded by the UK E-science programme, but rather, by a national translational cancer research programme. Secondly, all of the project partners were located within the same city, and the project had a colocated office[57]. In fact, the decision to colocate was deliberate in order to reduce the complexities associated with distributed teams (TransProject research proposal).

7.3.2. The Complexity of the TransProject System

On the surface, the system needed to support the TransProject activities should be relatively simple. The core of the system is a database storing the cohort (dataset),

[56] In fact the project team struggled to get any substantial access to the PIs, who were the ones conducting the scientific studies on the cohort collected. As clinicians, the PIs had cancer patients they were treating (and operating) on, and they prioritised this aspect as the most important part of their work. The team members recognised and understood that they were not the highest priority. On one occasion, one team member commented about the PI's lack of attendance at the team meetings, that (paraphrasing them): given the choice between saving someone's life and being in a TransProject meeting, it's a pretty simple choice.

[57] Although the two e-scientists were based away from the colocated office, the team had weekly meetings and both had the option to work within the central office.

starting with simple fields such as demographics (age, sex), administrative and audit information (address, GP, consent obtained), and cancer-specific data (tumour type, risk of breast cancer). Previously, in CancerStudy (a similar study to TransProject), this information was stored in a database, over several different tables. Patients (or cases, as they were known in the system) were each given a unique identifier across all the tables[58].

The local university's genomics and proteomics facility were expected to process some of the tissue and blood samples taken from the patients[59]. It was expected that one of the first steps in developing the system would be to establish the data links between the TransProject computing infrastructure and the local proteomics and genomics processing facilities. Initially, in the research proposal, the intention was to use Grid technologies to create a special link between the proteomic facility and the TransProject system. Using Grid middleware seemed to 'fit' well with the problem at first, since the data generated from the analysis was expected to be in the order of gigabytes of data per sample. If it was likely that there would be a substantial volume of tissue samples to be analysed (and by the time fieldwork ended this was still not clear), then terabytes of data storage might be required. It was expected that this data would need to be transferred into the TransProject infrastructure. There were naturally concerns raised regarding how best to manage the bandwidth required to be able to handle the volume of data once the system was up and running. However, it became clear, much later on, that there would be serious reservations about this option because of concerns about transferring patient-identifying data, as a result of the perceived inadequate security controls in Grid middleware.

---

[58] Note that in the system, there is a need to store administrative data, as well as the 'scientific' data.

[59] Although it was not certain how many samples would be analysed since the costs of proteomic and genomic testing is prohibitively costly.

As mentioned previously, part of the dataset to be collected would be generated by, and obtained from, external sources such as the Scottish Cancer Registry and local laboratories. It was proposed that TransProject would build an electronic network link with outside organisations in order to transmit this data electronically, rather than through physical artefacts, including paper or floppy discs, which were the existing channels of communication. In order to do this, it required negotiations with the IT departments (and their superiors) of several NHS organisations to allow having an added part of the system that would export data on a regular basis. This would require an in-depth understanding of each organisation's current IT infrastructure so as to reassure all concerned parties that the data was being released into a trustworthy and secure environment[60].

Another aspect of the system was the access to computational and data resources which would facilitate data analysis. Analysis tools could include: data visualisation, data searching or running data analysis packages. Part of the reason for the vague description of this aspect is that there was an understanding that data analysis tools would be developed in the final stage of the infrastructure development. Thus, it was only ever discussed in high-level terms.

Developing the system to store and process the cohort was only one aspect of the project. Another significant part of the work was to decide on and implement the processes and SOPs needed to collate the cohort. This aspect of the project involved an intricate understanding of the purpose of the scientific studies for which the data would be used i.e., the type of hypothesis that the data would need to prove or disprove. For instance, questions relating to whether controls (i.e., people from outside the population of patients) are to be recruited on to the project will be

[60] Part of the tactic to do this could be through the use of better-established technologies such as Java, rather than using Grid technologies. Thus, a substantial amount of work was involved in negotiating with the external organisations what data should be released, and how often.

dependent on the types of comparisons the oncologists would want to make in the future. Getting the SOPs correct turned out to be a particularly critical and difficult task. The SOPs, in particular, were important to get 'right' because they affect the way in which the tissue samples are treated: e.g., should the tumour tissue be prepared before or after it has been frozen? Such differences in preparation lead to significant differences in the quality of DNA obtained from the samples so that they cannot be compared as the same. As the Project Manager noted, "...if you change the protocol half way through, you have to throw the first batch away because they can't be compared with each other" (Fieldnotes, 21st June 2005). In short, a significant proportion of the project involved working out the procedures and processes for collecting data.

The project team not only had to design the procedures to collect the data, but also the computing system to process and store that data. At first, this feature of the project might seem to be a useful characteristic, because the system could be designed to effectively support the procedures to help collect the data, and conversely, the processes could be designed in such a way as to accommodate new technologies which might not be possible with a specific predefined system. However, rather than providing a source of flexibility, this feature in fact, proved to be a further source of ambiguity as there were ongoing debates (sometimes heated) about which was the priority – the processes or the system. The developers argued that it was more important for the processes to be decided first as they could then develop the system to fit around that. However, the Project Manager (who was responsible for designing the processes) argued that it was more important to have a functional database system for them to enter the data in first, as it was important to have a prototype/pilot study to demonstrate progress.

In the next section, the story unfolds to show the challenges of working out requirements for the TransProject system. Understanding the different choices available to members of the project team, when they ran into a particular constraint, and how they assessed those choices, requires an understanding of the science

(cancer research), the practical work of the hospital, and the capabilities of the enabling technologies they chose to implement the system.

7.4. Studies of the Practicalities of the Developers' Work

Button and Sharrock (1998), in their study of the software developers at a photocopier manufacturer, characterise the ambiguity of software development work nicely:

> *"One way in which software engineers work out their problems, and reach solutions, is in the very course of writing the code. But this practice carries the risk that code-writing will proceed without adequate preparatory analysis: the engineer may start working on the code as soon as the problem is recognized [sic], rather than subjecting the problem to prior and systematic investigation. It may then turn out that the identified fault is, so to speak, only a symptom and not the disease: the engineer's solution may merely reflect a superficial understanding of the problem. To address this possibility, the technical community has devised ways of delaying code-writing until after an adequate formulation of the problem. 'Requirements analysis' has been introduced to ensure that a methodical, systematic analysis of a problem is undertaken before deciding (a) how it is to be solved, (b) what is sought from the technology, and (c) what needs to be done to deliver exactly what is required. The engineer, unless subject to such constraints, may work with only a 'rough and ready' conception of the problem, seeking to clarify its nature during code-writing." (Button & Sharrock, 1998, p. 93)*

Button and Sharrock's characterisation of development, to a large extent, is applicable to most software development situations. Problems are voiced; developers gain some understanding of the problems and how they mesh together; code is written; and the developer's idea of whether they have characterised and understood the situation accurately is clarified, through feedback from end-users, other developers, or management. In working out requirements for e-science technologies, the situation is not that different: developers still have to gain an understanding of the problem that the software has to 'solve' and they still have to write code.

7.5. Difficulties in Finding a Feasible Solution

It seems often to be the case for e-science that there is a degree of uncertainty about whether the developers have gained a good-enough understanding of the problem at hand. Technologies have to function across different organisational boundaries and within different knowledge domains, requiring team members to grasp an understanding of a number of diverse disciplines. In designing the TransProject system (and this includes the broader data collection processes that the team also had to include), the project team had to sufficiently understand cancer research, the practical work in hospital wards, and the characteristics of the enabling technologies. Not only did team members have to develop a good understanding of the different domains, but also the dynamics of the organisations hosting those domains. Thus, the development team had to grasp knowledge of the particular cancer research that the TransProject PIs (oncologists) were conducting, the work practices of the hospitals that they were going to be collecting data from, and how particular technologies are favoured by certain organisations[61]. In short, team members have few assumptions that they could reliably make, resulting in every decision being drawn out into an extensive information-finding exercise, to ensure that they have a good-enough understanding of all the factors before attempting to reach a decision as to how to proceed. In addition, the project team had to do this with an eye to the dynamic nature of the organisations they were partnering: anticipating the changes (technical, regulatory, organisational) that were in the pipeline, and what implications these choices would have for the decisions the project team had to make.

---

[61] For instance, it was often commented upon that NHS computers run Windows, and that NHS practitioners preferred to pay for software because it afforded them system support (i.e., when it broke, they could call someone). However, in software development, and in particular, in academia, open source technologies were often favoured. Although it never became a serious issue, this difference in cultures did mean that the project team had to discount particular solutions that they had in mind.

In this section, attention is drawn to two empirical examples that allow us to explore some of the issues detailed above. These two examples were taken from a workshop early on in the project when only two members of the team were working full time on the project[62]. The workshop itself was called by one of the developers who wanted to bring together the different stakeholders of the project to share information and to discuss the progress of the project, and to settle on certain options previously talked about. The agenda included the composition of the dataset and the link with the local Genomics facility. As well as information gathering, another reason to bring everyone together was to obtain consensus on several important matters, including the dataset to be stored.

The first example concerns an apparent resource constraint in the data collection process; in particular, a lack of human resources to collect a complete record for each patient. The second example concerns the collection of the family history for the patients.

Example 1: Talking through Potential Solutions

Collecting patient data on the ward is often one embroiled with tricky ethical and practical considerations. Recruiting patients into the cohort not only requires an understanding of the emotional state of the patient, in order to assess the ethical stance of the nurse's actions, but also the implications for the cohort itself. Consequently, during the design of the TransProject study, these practical concerns had to be grappled with, in addition to understanding how these concerns would play out in terms of implementation choices for the system.

[62] The Project Manager had been selected, but had not started the job; the database manager had started the job about a month previously, and the developers had been chosen. One developer had been in place for seven months, while the other developer was still working for another project, but made herself available for this workshop.

The following fieldnote extract was from the TransProject workshop towards the start of the project (as mentioned previously). The exchange was between the Project Manager, a CancerStudy representative, and one of the developers. CancerStudy was another colorectal cancer cohort accrual project, targeting a slightly different population, but which had developed all of the initial tools that were expected to be part of TransProject.

The main point to look out for in this fieldnote extract is the sense of ambiguity involved in defining exactly what a feasible solution looks like. The extract starts with a discussion of the problem (insufficient nurses to collect the volume of data desired by the oncologist) and several proposed solutions (take on more staff; reduce the dataset to be collected). However, even though several options are considered, a clear winner never emerges:

> ***Project Manager****: "We have to take on more staff, or we take on less people for the study, or we do less questions."*
> ***CancerStudy Rep****: "You'll not get away with doing less patients – he [PI] will not miss 5 patients"*
> ***Project Manager****:" that's a decision to be made by the PIs. We'll see how it goes."*
> ***CancerStudy Rep****:... "I've worked three and a half years with him, and I know the effort he'll go to, to backtrack to get people that we've missed. People that we miss on the wards. There's retrospective recruiting as well, how are you going to deal with that, because you are going to miss people on the wards."*

There seemed to be an acknowledgement that collecting the different data would be too much work for the nurses. The Project Manager starts by suggesting financial or managerial measures i.e., "more staff", or reducing the scope of the study "less people", or "less questions" (i.e., data). The CancerStudy Rep, then makes the case that having less patients is an inviolable constraint for scientific reasons, using her personal knowledge of working with the same PI to substantiate her claim.

> ***Developer****: "well are you? If its just one hospital?"*
> ***CancerStudy Rep****: "Yep. You might have somebody that comes in from [Neighbour hospital]. There's some complications happen over in [Neighbour town] and the surgeon goes off sick, in [Neighbour town]. They have to be quickly shipped through to*

[TransProject City] for an emergency, they're maybe only here for 3, 4 hours and then back again ... "

Also, the recruitment process had to occur as soon as possible once a patient entered the oncology ward. Since many patients were elderly, the nurses acknowledged that some of these patients might not survive the surgery. A longer gap between admission and recruitment of a patient could bias the population: according to one of the PIs "... the longer we leave it [the patient recruitment], the more biased it is because half of them will die" (fieldnotes, 30th Aug 2005). In addition, if the nurses did not get consent before surgery, then it would be unlikely that they would be able to get hold of the tumour tissue, another important part of the dataset to be collected. Added to the complexity of the situation, each hospital would have their own particular working practices that had to be taken into account.

***Developer**: "Where do they go for their follow up? Do they not go back to [TransProject City]? Do they not talk to the surgeon that actually did the operation?"*
***CancerStudy Rep**: "No I don't think so ... I just don't think you can go for the whole lot"*
***Developer**: "That's exactly the type of problems that are still up there in the sky. There are all these problems that still exist - which is exactly why we need to do things like this [the workshop]."*

The developer commented on how these problems are "still up there in the sky", reflecting the fact that the requirements were not yet fixed. This captures the sense of ambiguity of the solution that they were trying to work out. Perhaps an even more insightful part is how the developer noted that's "why we need to do things like this" – referring to the workshop; this points to the important role of face-to-face team meetings (as discussed in the eMinerals chapter) and in general, the need for multi-disciplinary discussions to expose aspects and assumptions which other members had not thought of. For instance, the CancerStudy Rep drew the attention of the TransProject team to "retrospective recruiting". The team's reaction (probing about whether retrospective recruiting was a 'fringe case') served to point out the team's somewhat naive understanding of the practicalities of collecting data on oncology wards.

> ***CancerStudy Rep:*** *"It's obvious that we don't have the funding or the staff to do the full work ..."*
> ***Developer****: "Well that's the thing. So we might end up with data prioritised. This is what's always collected, and this is what's collected if there's time and the staff to do it. But in the end, the database needs to cope with all of it. So, the database needs to have all those fields and needs to make all things possible, even though most of these values are going to be null. From my perspective, it doesn't make much difference whether it gets done or not. The question for me is, and I'm being very selfish, what do I need to cater for? Or we can be selfish together, what do we need to cater for?"*
> ***CancerStudy Rep****: "To me the beauty of the [TransProject] system is the things like, the cancer registry, the death records. All these things that I can get without sending out a form to get that data ... just to be able to pull that data in. To me, that would be fantastic ..."*

Another option around the problem was suggested by the Developer: "we might end up with data prioritised". This suggested a compromise in differentiating the priorities of the various pieces of data – of course, without the PI present at the meeting, it is not clear whether this would have had serious implications for the scientific aspect of the project.

The CancerStudy Rep then seemed to suggest another alternative (one that is more technically driven) i.e., the possibility of automating the data acquisition process: ("all these things that I can get without sending out a form"). The implication being that such an automated process might free the nurses enough to collect all of the other data.

Another interesting part in the extract is how, towards the end of the extract, the developer tried to ground the discussion in terms of what the implications might be for the system, and how the system might be configured to help. The developer points out that the 'when' and the 'how' of the data collection did not matter to them, but it is clear in this context, that it would have serious implications for the scientific usage of the system. In other words, an aspect that might not be important on one side, is important to another. Through verbalisation, the developer is trying to obtain

clarification and an implicit agreement from the team regarding the requirements for the database.

> ***Project Manager**: "That's one of the things that the database will have to have. That flags things – that there's something missing. So that those records will show up and say, look you have to collect this data etc... cos like, that would be powerful. The end of each week, or at the beginning of each week, that it flags that they were missing something – and it changes next week, so that the ones that have been completed are gone. And that way, you can keep track of things."*
> *(Fieldnote transcript, 11th April 2005)*

Throughout the above exchange, there was a sense that no solution was particularly favourable compared to the others or that one would emerge as the 'best' solution. It seemed quite uncertain as to what compromises the development team would have to make when selecting one solution over another. While Button and Sharrock (1998) note that software developers are often aware of the constraints that they operate under and recognise when there are certain solutions which are "engineering-solutions-that-we-can-afford; engineering-solutions-that-we-can-implement; engineering-solutions-that-we-have-time-for" etc: these choices are often not apparent at first. Instead, team members have to take time to learn about the potential implications of particular decisions by talking about the problems, discussing the various options and by doing so, learning to get a feel for what could be possible, what would not be possible, and where it would be possible to make compromises. Thus, even though a solution was not settled on, this exchange served to clarify the problems and identify the possible constraints in the situation; part of which involved appreciating the interdependencies across different parts of the system. The lack of PI input is critical here, as this made it tricky to decide on various issues without the relevant expertise to hand. All this characterises the complexity of the situation that the team members find themselves a part of.

Transproject initially had to support research on colorectal cancer. The causes of colorectal cancer are often associated with lifestyle and food choices throughout one's lifetime, and its sufferers are often of the older generation. Hence,, the cut off age for CancerStudy was 55 years of age. In contrast with this, Breast Cancer, the other type of cancer that TransProject focused on, is often linked with genetic causes instead, its sufferers are mostly women, and there is a wider age spread for this disease.

Several concerns arise due to the typical characteristics of colorectal cancer patients. For a start, the patient may be in a level of shock, as typically they would have learnt of their diagnosis only days before, and have been scheduled for surgery soon after. As a result, the patients have to be approached with a level of tact and sensitivity. Another concern when approaching a patient is that the majority of these patients are often elderly, and when obtaining consent from them, the nurses have to be very careful to ensure that the patients fully understand (as much as possible) what they are consenting to. This may include their blood and DNA records residing for years in the TransProject system. Nurses may employ a variety of strategies to ensure (and account for the fact) that the patients do not feel pressured into participating in the study, such as speaking to them on admission, leaving forms and literature for the patients to read, and then coming back the next day to answer any questions and get their consent.

Another part of the data collected is the food and frequency questionnaire. This is generally a document upwards of 30 pages that contains detailed questions of foods eaten, and general lifestyle questions related to smoking and exercise. It is generally accepted, especially with elderly patients, that either the nurse has to go through the questionnaire with the patient, or if they give the questionnaire to the patients to take home, they have to spend time explaining it while developing a rapport with them in order to increase the return rate. The nurses will also have to take a family history from the patient, and it is likely that this would be done during the patient's stay at the hospital. Both documents are retained afterwards for analysis purposes.

Example 2: Improved Understanding of Problem changes Solutions

The second example presents a similar scenario. But this time, of particular interest is to point to the apparent fragility of the formulation of a 'good' solution. To continue on the theme of complexity of requirements, one interesting situation is that even in circumstances where the project team thinks they have arrived at a workable solution - we definitely need *that* functionality – it is only done so in the context of their current understanding of the problem. In the next extract, the discussion in the workshop had moved to the storage of the family history data.

Family histories play a vital part in cancer research, as they serve as a mechanism for being able to assess the risk of a family member developing cancer. This is particularly important in cancers that have been traditionally linked with genetic causes, such as Breast Cancer. In the extract below, the conversation had moved onto how to best to store family history data as it has to be formatted in a certain way in order for it to be useful to the oncologists. In fact, the CancerStudy Rep, earlier on, told their 'cautionary tale'. The study had initially collected the family histories on paper, and after having collated thousands of them, found themselves in the unfavourable position of working out how to transfer all of this data into an acceptable and analysable format for the oncologists to use. On hearing this, the TransProject team accepted that this could potentially be a problem for them, followed by a further discussion about the volume of work generated by the family histories in the existing process, as well as some of the practical and ethical concerns that the CancerStudy team had to deal with when asking patients for their family history. The problem is similar to that seen in the first extract; the CancerStudy Rep, with her experience and knowledge of the situation, argues there would be too much

work for two research nurses[63]. At the end of this exchange, one of the developers tries to draw out some implication for the development of the system; he points out that one of the potential solutions to leverage this resource constraint is by reducing the number of times that the data is recorded:

> ***Developer**: "I guess the thing about it is that it's recorded twice – once on paper and then once on system, the question is whether for [TransProject] we can directly record it onto the system. Because it is something which is collective worked up by the nurse and the person involved"*
> ***CancerStudy Rep**: "... putting someone's family history onto a computer is different for someone when its drawn onto a piece of paper beside them. They'll say what tv programme do they want, what size is your waist – they'll give you all of that information to put into a computer and not bat an eyelid, but taking down a family history onto a computer, not sure how well they'll do on that one"*

The suggestion here is that by recording it electronically – rather than via paper – they could reduce the time and effort required for the recording of the family history. Later on in the project, it was also pointed out that by doing so, a custom piece of software could be written so that family histories could be validated 'on-the-fly'; thus, reducing possible errors in recording (and therefore, potential re-work for the nurses). In this context, there seems to be a strong pull towards direct electronic entry of the information as a solution. However, the CancerStudy Rep points out that whilst that might solve the issue for the nurses (they would not have to record the data twice) and oncologists (because they would have access to the information, more quickly), the CancerStudy Rep raises concerns that this solution is unlikely to work in practice. This is because the patients are likely be made more aware that their family histories will be electronically by having them entered electronically. Therefore, there is a sense that even when discussions headed towards a favourable solution, further constraints might arise which render a solution untenable.

---

[63] In addition, it emerged later on in the project that, as well as the nurses simply recording the family histories, they would have to be trained to 'code up' the family histories, in order to assess the risk of family members.

***Developer**: "you mean because they've concerns about the"*
***CancerStudy Rep**: "Its a very personal kind of – it's a whole new ball game giving that sort of information"*
***Developer**: "Don't they understand it's going into a computer anyway?"*
***CancerStudy Rep**: "No"*
***Developer**: "Isn't that unethical?"*
***CancerStudy Rep**: "It was never intended to [be entered in electronic form]"*
(Fieldnote transcript, 11th April 2005)

Perhaps in an ironic, yet symmetrical situation, the fieldwork extract ends with the CancerStudy Rep pointing out that they had never intended to have to enter the family histories electronically, because they did not know that the data had to be stored and formatted in a particular way in order for it to be useful. This illustrates the 'missing middle' point further, by showing how CancerStudy themselves had similar issues of learning at the outset.

Because all these considerations have to be taken into account it is very difficult for the development team to work out requirements *a priori*[64]. Amongst all the domain information, a working understanding of partnering organisations, and the future trajectories of the partnering organisations the team members have to find a workable solution. It is often uncertain which solution is 'at the start and these solutions are often based on preliminary opinions about the characterisation of the problems and the solutions. Button and Sharrock (1998) call this realistic solution implementation the need for software developers to "make do" in formulating what can be realistically achieved. However, in e-science, it is not so much that team members have to accept a realistic solution, but sometimes, it is not clear what a realistic solution is: *What is a sensible solution to this problem? If we make this decision, what are its implications elsewhere? Are we happy to accept that something else?*

[64] Or carry out a thorough one-time requirements analysis and on top of this, as mentioned in AstroGrid chapter, different organisations will progress at different rates. Thus, there needs to be a management of the solutions across the boundaries.

What are the movable and immovable constraints? What parts can be compromised, and what cannot? Is there any other way around it?

In the end, the workshop did not meet its goals in establishing agreement because the PIs (who were responsible for making the final decisions) could not come to the meeting due to time constraints. Thus, very few decisions could be made at the meeting. In fact, if anything, the workshop lead to frustration rather than resolution because it seemed to raise a lot more questions about the system and served, to an extent, to complicate matters as project members further grasped the complexity of the system that they had to develop. Two empirical examples stood out as demonstrating the intricacies of working requirements out; both concerned similar scenarios: a problem was identified and the discussions revolved around a number of different solutions in addressing the same problem, but no single option emerged as most favourable.

In summary, e-science developers often run into uncertainty regarding how best to proceed, and what assumptions can be safely made about the situation. In the next section, the focus is shifted onto using empirical data to explore the issues outlined above.

7.6. Strategies in Stabilising the Solution

From the above examples, articulating requirements for e-science technologies seem to be difficult at first, because a substantial volume of information has to be assimilated to make an 'informed' decision, with only a partial picture available. Each project team member is attempting to grapple with the problem-at-hand and make sense of how different requirements fit and mesh together into a functioning whole. Over time, a fuller, more informed picture is slowly built up as the problem is further clarified and possible solutions narrowed down; requirements are not simply elicited, but are worked up over time (Bergman, King, & Lyytinen, 2002a). In the

context of an e-science project, this process could possibly take longer, because a feasible solution is often contingent on considerations from several domains of knowledge and thus, might require a longer exploration period.

Over the course of working out requirements, parts of the system begin to get agreed, consensus emerges, and solutions get *stabilised*. In this context, the concept of stabilisation is similar to the meanings found in information systems literature (McLaughlin, Rosen, Skinner, & Webster, 1999) that describes 'how things come to be'. Stabilising a requirement is about getting a working agreement on how to proceed, given the information available at the time: maybe further information is needed, maybe a direct decision is needed, or maybe the requirements have to be formally represented in the form of a requirements specification document.

Because of the large degree of uncertainty that exists at the start of e-science projects, it is important to consider *how* this stabilisation occurs. During the fieldwork, five different strategies to stabilise requirements were noted:

- Stabilisation by discussion
- Stabilisation by building consensus
- Stabilisation by producing an artefact
- Stabilisation by precipitation of a decision
- Stabilisation by formal representation

These strategies serve to highlight the variety of ways in which elicitation and agreement techniques are employed in navigating the complexities of working out, and getting consensus on, a set of requirements. In this section, I will briefly describe each of these strategies and illustrate them with an example of how they were used to stabilise one particular requirement: the dataset to be stored in the system.

7.6.1. Stabilisation by Discussion

One of the most common methods to enable stabilisation is by talking or simply discussing the issue with the team members. Being able to 'have a chat' about the situation served as a way of understanding the situation, throwing out potential ideas, and formulating solutions. Through these discussions, a better picture of what the situation is and what options are available is built up. Whilst sometimes, these discussions may occur in formal settings such as in the Tuesday morning project meetings, they often happen in more informal situations. In particular, being located in a central office facilitated this process: team members would often turn and talk to other team members to ask them what they thought of their particular situation, sometimes bouncing ideas off each other. There were times where someone would run into difficulty whilst coding, and having other team members to look over what they had implemented so far, served as a way of prompting a discussion, leading to clarification of the situation and proposal of suggestions.

All this, points to the way that the project team begins to get a better feel for how certain solutions might be more difficult than others, or that certain organisations might be more responsive than others, and eventually learn to appreciate the interdependencies across the system they are developing. For instance, the two examples in the previous section serve as illustrations of how - by talking through a problem, proposing possible solutions and seeing what objections arise – the whole project team can begin to build a clearer picture of the situation. More importantly, team members use this knowledge to anticipate possible issues in future work and discussions.

In the context of agreeing the dataset, several emails and informal 'chats' were exchanged towards the start of the project between the oncologists and the full time developer on the project. These discussions seemed to be tentative and the exact list of data that needed to be stored or collected in the system was unclear. During these early discussions, the developer visited the CancerStudy team (fieldnotes, 10th Feb 2005). They spoke about the CancerStudy dataset and how the data was collected. It

became clear that there were certain data such as family history and blood samples that were likely to be in the system, but it was not clear what additional data was to be collected in the TransProject, or indeed what it was not necessary to include. Thus, it was decided that the current CancerStudy dataset would be placed on a wiki, and the oncologists were asked to write comments there, in order to have a centralised place to exchange the discussion regarding the dataset required (as an alternative to email).

7.6.2. Stabilisation by Building Consensus

Another strategy to stabilise requirements was to call meetings and workshops. This strategy is normally used to gather key stakeholders and decision makers in one place to get a clearer understanding of the issues, and often, to also agree the next steps to progress. In TransProject, this strategy could be seen in the project workshop towards the start of the project, and then later on, once the Project Manager was recruited, the weekly project team meetings.

After the dataset was added to the wiki, there was little indication that the oncologists were engaging with the feedback process. Those involved around the project were aware that the oncologists were busy individuals with a demanding time schedule; hence, the strategy from the team members was to call a project-wide workshop in order to gather key stakeholders in one place at the same time. The aim of the daylong workshop was to discuss the dataset (as well as other components of the system), and hopefully to agree on what the dataset would be. In doing so, they hoped to be able to get a consensus from the senior management clarifying the exact data to be collected, and where and when that data was to be stored. However, as outlined earlier on in this chapter, none of the senior stakeholders attended the meeting, citing time issues (fieldnotes, 11th April 2005).

The success of this strategy was dependent on sufficient senior representatives of the stakeholder groups together agreeing and enacting decisions. This could be

significantly more challenging in e-science, since the e-science project might not be a major part of the senior stakeholders' work (cf. users in participatory design where the engagement process is with end users whose work would change as a direct result of IT system implementation), and/or the stakeholders may be geographically distributed across a city or a country.

Another instance of the consensus building strategy was through the weekly team meetings. These meetings served as a regular 'check-in' point on the progress of the different aspects of the system[65]. For the team members, having this organised rhythm in their project work, crucially provided the team with an opportunity to inform the whole team about their progress, propose their next actions and obtain a consensus that they were 'on the right track'.

7.6.3. Stabilisation by Producing an Artefact

Another mechanism for stabilisation is to produce a physical artefact. The most obvious artefact to produce is a prototype of the system, and as Ure et al. (2009b) mention, an artefact serves as an effective mechanism for user engagement, even when the artefact does not accurately reflect end-user needs. Producing a physical artefact facilitates the stabilisation process by a) clarifying the requirements through a physical representation of the current understanding of the problem and solution, b) provides a probe into the feasibility of a solution i.e, allowing the developer to build up an understanding of the difficult aspects involved when implementing the full system, and c) provides a sense of progress in a project.

---

[65] This stabilisation technique also worked well in the AstroGrid project, where the entire team gathered, and work for the next iteration (cycle) was discussed. These regular team meetings served to discuss work from previous iteration cycles and build the recommendations for the next plan through discussions and coming to an agreement as to what the workplan for the next iteration should be (Interview, AG Technical Lead, AG/3).

Returning to the dataset example, a few months into the fieldwork, members of the project team were increasingly frustrated at the lack of commitment from the oncologists regarding the set of data to be stored in the system. In discussing potential actions to take in order to get an answer on the dataset, the Project Manager made the case that the developers should build a prototype, and in doing so, the oncologists could use the prototype as a platform to spot 'missing' data that they might require (fieldnotes, 30th August 2005). Consequently, one way in which artefacts serve to facilitate stabilisation is by stimulating clarification.

Eventually, the prototype became part of a large pilot study to test the entire system before the project went 'live' for collection. As part of the study, it was deemed that 'real' data should be used in the study: real datasets often contained missing data, and having the test dataset was an opportunity for the project team to test out their processes for dealing with data curation. As a result, a small subset of anonymised data was given by CancerStudy to be imported into the TransProject system as part of a test dataset (fieldnotes, 20th September 2005).

The pilot study also provided a mechanism for the data collection processes for the dataset to be tried out. Building the pilot study required the SOPs for collecting tissue samples to be defined. This shift was partly due to the ongoing debates regarding how data was going to be collected when discussing what data needed to be collected. Earlier on in the project, at one of the Tuesday weekly meetings, the subject of SOPs was brought up and one of the developers suggested a pilot study would serve to highlight "missing" SOPs:

> *Developer: "... Is there any progress on pilot study? Maybe we could put some SOPs in place?* ***If there were some missing, then we know what we need****" (Fieldnotes, 26th July; emphasis added)*

From the above data extract, it can be seen how building a prototype can force certain decisions to be made and can stabilise the requirements further, by stimulating clarifications or pointing out 'missing' parts of the system.

However, before leaving this discussion, it is prudent to point out here, that there should be an explicit agreement beforehand regarding the purpose of the prototype. Without going into full details, one illustration of this point was connected with the aforementioned prototype. One of the developers had produced a six-page basic web-based system that was a series of pages where the data could be entered into a new patient record, and the patient record could be retrieved. When the developer showed this system to the Project Manager, the manager started by clarifying what he thought should be the layout of the system, to which the developer replied "to be honest the design of the screen is not the main concern, we haven't got the dataset yet" (fieldnotes, 25th Oct 2004). This point ties in with the earlier discussion in the AstroGrid chapter about how having explicit development processes provides team members with an explicit indication as to the level of granularity of the requirements considered at that time.

7.6.4. Stabilisation by Direct Questioning

Another mechanism to stabilise requirements was through directly asking questions to force decisions to be made, and to get a commitment on those answers. Direct questioning is obviously an effective strategy for agreeing and eliciting requirements; but inevitably this could only be done after a substantial level of clarification regarding other parts of the system, by the project team members who were not oncology experts,

After months of being unable to progress because the dataset had not been agreed, team members issued an ultimatum to the oncologist (and Lead PI), arguing that they could not progress with the system development until a list of questions were answered. These included questions concerning the cohort population, which data sources should be trusted over others, and where the data collection should take place. Altogether, thelist consisted of 18 questions (with the most relevant being):

1. *Eligibility criteria [for the study] (all ages, treated in [city])*
2. *Informing GPs*

...
4. Food Frequency Questionnaire [sic]
a) Filled at home.
b) Need to send reminders
c) Need to send back to patient for repairs
5. Cancer and Lifestyle Questionnaire
a. Filled in where? At home or with nurse?
(fieldnotes, 30th August 2005)

The response was rapid. The oncologist attended the following team meeting and answered each question one by one. Most of the questions were answered by a yes or no, although some provoked a debate and then a decision was made within the team regarding that item. The ability of the team members to ask direct, specific questions in order to force the oncologists to make decisions demonstrates the volume of learning that the team members did, and how much of a grasp of the situation they had gained.

7.6.5. Stabilisation by Formalisation

Finally, requirements can be stabilised through formal representation – through documentations or procedures. Requirements come to be stabilised in this way because stakeholders are asked to 'sign off' on some formal account of what the requirements are. At this point a shared understanding has usually been reached within the team regarding the solution. The document can be used as an artefact to discuss and to further elicit requirements until an agreement can be reached. This is normally followed by a shift of emphasis to development and once the document has been signed off, it is unlikely to be contested to a great extent.

It is important to note that in order to be able to get to a stage where a concise formal account can be given, there is a prior need for a sufficient level of detail being worked out regarding what the system should be like. Thus, although writing documentation seems to be an easy way of stabilising requirements, a substantial

amount of discussion has to occur beforehand for documentation or procedures to be workable (and stay that way) in the long term,.

Over the course of agreeing the dataset, a greater discussion of the data infrastructure for the entire system was prompted. One particularly tricky issue, especially working in an e-health related system, was with respect to which information should be stored on the NHS IT system, and which information and components should sit on the local University's network. Although it was clear that patient identifying data had to sit on the NHS side, and that the large volumes of data generated by the proteomic and genomic analysis would have to be on the University's network, there were on-going disputes regarding the exact arrangements of which computers or storage devices should be on which network.

After weeks of discussion, one of the developers wrote a document with diagrams depicting where each component sat (NHS or University), and detailing the current and future phases of the IT infrastructure (fieldnotes, 25th October 2005). It was effective because once the team agreed on the infrastructure outlined in the document, it served as a decision from which the team could move on and build upon. This might seem obvious and trivial, since, as mentioned in Chapter 3, the requirements specification document is a mechanism to facilitate such discussions between the parties regarding the requirements. In this context, where the requirements and solutions were ambiguous and often revisited, having a formal document served as a mechanism to ask for a commitment by the project team to a plan, and hence, stabilised part(s) of the system.

A related but side point, is that documentation served as a relatively effective mechanism to explain highly technical decisions to non-specialists in the team; team members seemed to respond and engage more easily in the discussion by having technical jargon, or complex technical ideas expressed in written terms instead of only in a verbal manner (e.g., through the discussions).

7.6.6. Stabilisation as a Working Agreement

Finally, before leaving this section, it is important to point out that while requirements and solutions get stabilised, these stabilisations only serve as temporary working agreements based on the team's existing understanding of the problem. If further information arises that further informs the team's understanding of the situation, or if the external situation changes, it might force a radical re-conceptualisation of the solution needed if the pre-existing solution is not deemed fit to satisfy the new understanding of the situation.

One example was during the middle of the observation period when a radical re-structure of the system occurred. During a discussion between one of the developers and the breast cancer oncologist regarding the dataset specifications, the oncologist mentioned to the developer that each patient could have multiple and recurring episodes of the same cancer. Up until this stage, the system architecture was built around a patient, and each patient had demographic data, tissue/blood samples, genetic data, family history data and details of the cancer related to that patient. The implicit assumption was that each patient would have a single episode of cancer. Hence, in order to be able to record recurring episodes, the architecture had to be radically changed to reflect this. From a patient-focused system, it had to be changed to an episode-focused system: where each episode would have a patient, rather than each patient having an episode. Although at the outset this may seem like a simple change to the oncologist, it required a substantial restructure of the system. Examples such as this illustrate how requirements which may have seemed stabilised, might become unstabilised as new requirements and information emerge. Thus, this process of stabilisation is ongoing throughout the project. Although changes due to emerging requirements are a well-established phenomenon, the suggestion here is that this might occur more often in e-science because it is often difficult to get a clear understanding of the whole picture due to the multi-disciplinary nature of the project. There might be a perceived high-level of uncertainty as requirements which were previously thought of as stabilised, later on become un- and then restabilised, as the conditions and constraints which the system has to operate under are further clarified.

7.7. Discussion

This chapter has detailed the everyday practicalities of requirements work in an e-science project. Of particular concern is to describe the practical concerns and considerations of project team members when grappling with the requirements of the infrastructure in development, as well as learning to develop an understanding of the solutions available, and to assess the consequences and implications of one solution over another. Through this detailed account, some of the difficulties in understanding requirements for an e-science infrastructure are explored: specifically, the complexities involved in being able to find a feasible solution.

In examining the details of everyday work in understanding requirements, it is possible to see the complexities of working across multiple domains of knowledge and multiple organisations. Not only do the developers have to get a working knowledge of the different domains (oncology, practical work in hospital wards, Grid computing), but also an understanding of the contexts in which these domains are situated (the type of studies the oncologists are interested in, the local oncology ward(s), the NHS IT systems), with an eye towards the future trajectories of these domains and organisations (the future studies that the oncologist might be interested in, possible changes to the working practice in the ward, the wider NHS IT infrastructure programme). The volume of learning involved should not be underestimated, as highlighted a number of times earlier on[66]. It is therefore unsurprising that the team drew on a previous similar study – CancerStudy – as a

[66] Another important point to make in this discussion is that the talk of 'learning' seems to imply the situation is one way, i.e., from the world to the team member; it is not the case that members are passive bystanders to the whole situation. Team members, themselves, can actively shape and seek to influence the future decisions of the partnering organisations. They can find the 'gatekeepers' and seek to influence the resources that they can release; they might seek to find opportunities to trade 'favours' with partnering organisations and so forth. Thus, the complexities of understanding requirements increase, as the possibilities of shaping future outcomes and decisions are possible.

resource in their understanding of the system. In addition to learning about other domains, to an extent, team members have to appreciate that part of their role is to educate other team members about their domain. One potential recommendation to quicken this process is better training for e-science participants through further multi-disciplinary science programmes such as bioinformatics.

In light of the above discussion, there is a strong argument to be made for the need to accommodate the explorative nature of e-science projects at their start. One possible approach to alleviating this problem is by scheduling in extra time for exploration towards the start of e-science projects. This is especially prudent when a new team is working together at the same time (echoing the argument in the eMinerals chapter, as well as one made by Lloyd and Simpson (2005)).

In following the team's exploration of working out requirements and potential solutions, one matter to consider is how requirements and solutions begin to solidify and settle, and thus, how requirements and solutions get *stabilised*. There are many steps in stabilising a requirement; starting with clarifying and expanding the nature of the problem, with some consensus being built up regarding the problems-at-hand; concurrently, potential solutions are discussed, objections raised and constraints made concrete, and slowly a shared understanding of the solutions is built up. In this context, stabilisation is a negotiated phenomenon; it is a fragile balance where a particular set of requirements is agreed, and feasible solutions are found. The balance can be changed when new information comes to the fore, or if the future solution space is changed in some way.

In the latter half of this chapter, the focus was on the different strategies employed in stabilising requirements and solutions. Examining these multiple strategies in practice serves to highlight the variety of ways in which team members practically navigate the complexities of the situation that they are faced with. In identifying these strategies, it might serve as a resource to future e-science participants when facing similar challenges.

Finally, from the fieldwork data (alongside a comparison with the AstroGrid and eMinerals case studies), there is a strong argument to be made here regarding the need for dedicated project scientists in e-science projects. In TransProject, the scientists were oncologists with very limited time and resources, and the team members were always aware the oncologists were busy individuals, with little time dedicated to TransProject system development. Even though the project was co-located (cf. distributed), the TransProject team seemed to find it more challenging to engage with their end-users compared with the previous two case studies. Thus, working out requirements relied on a commitment from the scientists, as well, and physicality of access (i.e., being centrally located) should not be confused with having access (in terms of time and commitment).

Chapter 8: Discussion and Conclusions

8.1. Introduction

This chapter provides an overarching analysis of the case studies presented in this thesis and discusses its implications for the wider e-science field.

The chapter starts to address the aims set out at the start of the thesis, i.e, to complement the existing literature by providing a much more strategic overview of requirements engineering challenges in e-science projects. There are two parts to the discussion. The focus of the first part will be a discussion presenting the concept of requirements as contested entities. The discussion is centred on the idea that articulating requirements necessitates a skilful navigation between the set of problems that e-science could solve, and the set of possible solutions that may fit. This navigation goes back and forth between the two spaces as they become more defined and the analysis picks apart the complexities of conducting requirements engineering work in e-science projects; reaching further than simple user engagement techniques. The discussion then returns to the framework of the challenges as presented in the literature review chapter (Chapter 3) and enhances it with findings from the case studies data. The chapter finishes with a discussion of the

implications of the broader conceptualisation for the existing literature and e-science practitioners.

8.2. Approaching a better understanding of requirements work in E-Science

8.2.1. Requirements as Contested Entities

At the heart of this thesis has been an examination of how the e-science vision is realised in real applications and infrastructures. These are often large-scale, distributed systems that have to support complex, highly-skilled scientific work. A combination of social, technical, organisational and political factors come into play in order for such structures to be a) produced and b) sustained. On one hand, there are the ambiguous interpretations of the e-science vision, and on the other, is the uncertain (and at times, volatile) environment of enabling technologies.

Throughout the studies, a key observation is that requirements activities are not obvious. Project participants openly acknowledged that they did not engage with formal methods prevalent in requirements engineering, nor did they have the discrete requirements gathering phase that is expected at the start of most software development projects. They also stated that they did not have anyone in the team whose only job was to elicit requirements, yet when questioned further, participants talked about the ways that they came to understand what needed to be built. More often than not, their accounts were similar to research activities. They would be given a description of what they had to build (sometimes with clear specifications, and at other times, simply a rough description) and their task was to research the 'best' approach to fulfilling the requirements of the component or application. A common phrase in the interviews with project participants was: 'we needed this component or application, but we didn't know how it might work, or how it feasible

it is to build, so we tried to build it and got this'. This pattern was evident across all of the cases – from building multiple components in AstroGrid, hinted in creating 'do-able problems' as outlined in the data and tools built in eMinerals, to the fieldwork extracts in TransProject of project participants attempting to grapple and stabilise requirements.

The process, as described by project participants, seems to suggest a blurring of the boundaries between what is traditionally considered as requirements work (i.e., building up an understanding of what the technology is required to do) – and what is traditionally characterised as "design" work (i.e., creating a solution to fit the requirements of the system). Viewed in this way, the process of articulating requirements becomes one of navigating between problems and solutions in order to find a satisfactory outcome (Bergman et al., 2003). Bergman et al. (2003) conceptualised requirements as navigations between the problem space (i.e., a set of anomalies which are considered problems that have to be solved) and the solution space (i.e., the set of possible solutions). In the beginning, a project participant will have some initial idea of what the requirements are for that component or tool, and at first, there might be several possible configurations of solutions. These possibilities could be something as significant as selecting which enabling technologies to use (e.g., investigating the solution space) or they could be minor configurations (e.g., the size of the server space). Upon building a prototype, project participants would realise the advantages and limitations of that particular technology (revealing more of the solution space) or reveal some deeper set of requirements (revealing more of the problem space), and present their findings back to the rest of the project team. This would create a discussion within the team regarding whether the characterisation of the problem or solution was correct, ultimately leading to a modified idea of what the solution could be (building on previous discussions). In the TransProject case study, the concept of stabilising requirements (section 7.6) was used as a way to describe how the articulation of requirements is one that is built up over time, through many discussions. What is important and what is not important is

slowly made sense of, in attempting to fit and mesh together constraints and parameters in a functioning whole.

An important implication from this conceptualisation is that there is a danger that it could take a substantial amount of time during the journey between starting with an idea of the solution, finding that not to satisfy all the constraints, moving back to redefining the problem and then shifting back to the solution. This to-and-fro takes time, and with it can come frustration, particularly if project participants feel as if they are not advancing as quickly as they should be. At worst, it is possible for participants to continuously navigate between the two spaces, never quite settling on a solution, with the added complication that the landscape and assumptions within which these spaces are situated also evolves with time. In the AstroGrid project where project participants adopted a formal software method, one can see that the use of the agile software methodology (3 monthly cycles of work), attempted to manage this danger to some extent. It did so by creating a temporal rhythm of development work with the creation of natural frequent decision points at which times requirements could be debated, stabilised, and built upon (Section 5.6)[67]. This general rhythm acted as an organisation mechanism by which progress could be measured (to an extent), and allowed participants to debate and understand the sorts of problems they were tackling collectively. During these sessions, project participants would employ a variety of strategies in order to bring some of those

[67] Another important dimension to note is that internal development progress was not the only rhythm that existed in development. A clear (and reasonable) second strand was the rhythm of accountability (Section 5.6). E-Science projects are large-scale projects funded by Research Councils. Naturally, funders would like to ensure that their projects are achieving what the original proposal stated it would achieve. Thus, a part of the project team's work is to demonstrate the progress that they have made as it moves along. Both in the AstroGrid case study and in the TransProject case, there were review points, and these existed to demonstrate the progress of the project both to those managing the project, and externally to the wider disciplinary community. This presented a pressure for project participants to progress and encouraged decisions to be made. Thus, this second rhythm, inside the development work, also helps the process of stabilising the project participants' shared understanding of what is required at particular points during the project.

contests to a conclusion, so that they could firm up and move on to the next set of decisions to be made. This pattern of monthly meetings to set frequent decision points was not confined to the large projects alone. In TransProject, which was significantly smaller, these stabilisation strategies were observed in the lead up to, within and after the weekly meetings. Five of these strategies were described in the TransProject case study: stabilisation through discussion, building consensus, producing an artefact, precipitation of a decision and formal representations.

8.2.2. Aligning Spaces to create 'Do-able' Problems

To backtrack a little, if requirements are the ties between points in the problem and solution spaces, a more significant question is raised: what makes a particular problem-solution pair demonstrate a promise of realising e-science? In more practical terms, if project participants are asked to build a tool, or a system, or a component of a system, what makes a particular pairing (however vague) show more promise than the potential tens, or hundreds of potential tools to be built? Hints to answering that question, can be found in the alignment discussion from the eMinerals case study (Chapter 6).

E-science projects typically start with a grand overarching vision of the sort of science that the project wishes to facilitate. These visions are typically derived from the grander e-science vision and reinterpreted from the perspective of that particular discipline. As a starting point these visions provide some indication of the problems or limitations of existing approaches, hand-in-hand with an indication of the technological tool(s) that will allow the scientist to work in an e-science manner. Once a project starts, the project team begins to narrow down and suggest ideas for technological tools that could be built, in order to realise this vision. It is at this point that the navigation between the problem and the solution space begins.

An interesting observation that emerges from the empirical work is the sense of how some scientific problems are amenable to fit within an e-science paradigm. Fuijmura (1987) argued that for a research problem to be considered 'do-able', it must be aligned on three levels: the experiment, the laboratory and the wider social world. In e-science, from the empirical work, there is some alignment work to be done. Team members in the eMinerals project mentioned how they found combinational problems, i.e., particular problems in computational chemistry that lent themselves to the Grid computing paradigm. These alignments have to work on three levels: between the scientific work, the characteristics of the enabling technologies and the organisational structure (section 6.5). By examining requirements from these three dimensions, it may be possible to narrow down the search for good fits between e-science and the scientific disciplines they seek to help.

8.2.3. Accounting for the wider environment and technological trajectories

Finally, drawing the last point into the discussion: the context and landscape of the e-science projects also play a significant part in creating uncertainty in requirements. In a discussion where requirements are seen as entities to be negotiated, it is important to point out that e-science projects do not exist in isolation. E-Science projects, by their very nature, cross multiple boundaries. These boundaries include the organisations that they are situated within, the disciplines that they are a part of, and the technological landscape that their underlying technologies (e.g., the Grid) have to sit within. One of the challenges outlined by the project participants is the need for their understanding of what are acceptable solution changes over a period of time.

Throughout all three of the projects, increased involvement in the wider community developed as the projects progressed; this was especially true for the resources that each of their solutions were built upon (e.g. a particular Grid component, the metadata standardisation body). While this was expected in infrastructure heavy projects, such as AstroGrid, the same applied to the smaller eMinerals project. For

both of these projects, the team wished to shape and contribute to the landscape within which they were situated. A major rationale for this was that by participating in these communities, they could be kept up to date with the 'trajectory' of these technologies. In this way project team members would be able to anticipate any changes that could significantly alter their view of what their requirements and the requisite solutions were. In its milder form, the eMinerals team, for instance, attended regular meetings in order to seek to influence the Grid infrastructure technology team. This helped them when they needed additional help in the running and maintenance of their own Grid. In the more "aggressive" case, the AstroGrid team sought to create the International Virtual Observatory Alliance, i.e., an international body that governed the data standards by which astronomical data would be shared. The team talked about how their metadata approach was significantly different from the researchers in the U.S. and their consequent need to lobby other countries around them, in order to make their approach relevant inside the astronomy community. In other words, project teams have to actively consider how changes in the wider environment can impact on their project, and whether they can seek to influence that environment. Part of the challenge in understanding requirements, therefore, is the need to appreciate the trajectories of the environments that the project is situated in, and to anticipate where the relevant technologies are going. In considering the wider environment, the aim of the project teams was to foresee whether changes in the trajectories would render their conceptualisation of the relevant problem or solution less useful.

In short, this thesis has shown that deciding requirements demands the process of skilfully (or not as the case may be) navigating between 'the problem' and 'the solution' to produce a realisation or instantiation (Ure et al., 2009) of the e-science vision. This process should take into consideration the feasibility of the solution, given the time and resources available, and raises such questions as: How do we turn the e-science/discipline/project vision into real life terms? What should its architecture be? What functionality do we require? Which databases do we need to connect to? How many users are we expecting to have? The role of requirements

work is to supply answers to such questions. Hence, the work of articulating requirements for e-science technologies is the result of constant refining and redefining the navigation between the problem space and the solution space. This work takes place while project participants grapple with how e-science could be brought into practice, and how to create the tools required to support these new methods of work. This is an on-going process through the entire project. As the participant's understanding of these spaces evolves, the landscape in which they are situated also evolves. This conceptualisation is consistent with the ideas of other scholars who have examined the wider issue of collaborative development in e-Infrastructures (Karasti, 2010).

8.3. General Challenges in conducting Requirements Work in E-Science Projects

8.3.1. Attempting to understand requirements work

Returning to the three case studies presented in this thesis, it is possible to see the challenges of bringing e-science into practice. Individually, each case study provides a window into the sorts of struggles and negotiations at play in the articulation of requirements for e-science technologies. It is at this point in the thesis that the commonalities between the different e-science projects are drawn together.

In the literature review chapter (section 3.2), a table was drawn up of the challenges of requirements engineering in e-science projects described in the literature. From the research described in this thesis, additional points, which might not have been highlighted before, can now be added to this table in order, to provide a strategic overview of these challenges. The aim of this section is to contribute to the existing literature by signalling some of the issues and tensions common to all three studies. Additions to the previous table fall into five main areas: no clear 'optimal' team; no

established requirements engineering process; mutual learning of domain knowledge; no clear feasible solution; and rapidly changing technological environment.

Characteristics of E-Science Projects • Geographically Distributed • Multiple and Diverse Stakeholders • **No clear 'optimal' team (8.3.5)** • **No Established Requirements Engineering Process (8.3.3.)** • **Mutual Learning of Domain Knowledge (8.3.4)**	**Population of End-Users to be supported** • Heterogeneous population of users • High autonomy of end-users
Characteristics of the work the software has to facilitate • Supporting scientific work • Supporting cross-boundary work • Creating transformational changes • **No clear feasible solution (8.3.2.)**	**Characteristics of Technologies used to enable e-science** • Poor usability and immaturity of grid technologies • **Rapidly changing Technological environment (8.3.6.)**

Table 6: Updated challenges associated with conducting requiremetns work in e-science projects

8.3.2. No Clear Feasible Solution

As discussed in section 8.2, most of the discussions with the developers seem to point to a similar scenario, i.e., they would be given a description of what they had to build; sometimes with clear specifications, and at other times, with simply a rough description. Here one of the AstroGrid scientists describes a typical situation that would occur:

> *" … people will say to me, ok, how does an astronomer pose this problem? I've been told that I need to build a tool to let some of you do such and such. What does this mean? How much are they going to use? How much data they wanted to be able to process?" (Interview, AstroGrid Project Scientist, AG/5)*

Not only did the developers have to do a large degree of research in understanding some of the tacit knowledge that the scientists had, they also had to work out the 'best' approach to satisfy the requirements. The situation was typified by statements such as: "*we need this component or application, but we don't know how it might work."*

In tandem with discussions with scientists on refining requirements, another challenge the developers faced was the uncertainty of what could or could not be feasible solutions. In the next data extract, the AstroGrid project manager explains the strategy of doing small investigations to see how much work there is to be done before committing to taking a particular approach. This presented one way of managing uncertainty and assessing possible solutions for feasibility:

> *"by taking these short iterations, what have you lost if something goes hideously wrong; you haven't lost a lot of time. Its a difficult thing sometimes to get across in management […] so I'm quite happy for a prototype for a couple of people to spend three months and then reach a conclusion that says this ain't going to work, because now I haven't wasted my whole team on trying to do it so an honest failure is useful information" (Interview, AstroGrid Project Manager, AG/2)*

In such situations, the requirements are entangled with design and implementation because the problems the developers are trying to solve are multi-faceted. Often the situation is much more akin to a research process. In all three projects, practically all project members were given a large degree of autonomy in what they did. This freedom is largely attributed to the academic and research culture within universities. It seemed rare for a comprehensive requirements list to be drawn up (or at least few references were made to such an occurrence). Instead, developers had to be content with producing potential solutions from a vague description of the problem along with timescales.

Lawerence (2006), for example, suggests that the tension between research and development occurs because of the reward systems that arise as a result of the composition of e-science teams. The computer scientists within projects have to produce something novel in order to obtain research results and write papers. The scientists, on the other hand, want a stable deployable system in order to be able to carry out their research. Perhaps one implication is that this situation is not as clear-cut as it may seem at first. It is not simply that the computer scientists or even the "professional" software developers want to introduce novelty into the system (although there are instances throughout this research where this could happen with strong incentives to do so). In fact simply framing the problem as a dichotomy between producing research and producing a rigorous developed system is somewhat unfair. Even in developing the e-science system, there is a large degree of research and investigation. Whilst Lawerence was addressing the problem of research versus development priorities within projects, it does not necessarily follow to frame the problem in this way. To paraphrase the earlier quote, producing e-science applications and infrastructure is not simply a case of arranging components together (Interview, AstroGrid Project Manager, AG/2).

8.3.3. No Established Requirements Engineering Process

None of the projects in the case studies had a formal requirements engineering phase. On being asked why this was the case, project participants were quick to point out the benefits of having project scientists on the teams:

> *" … one of the reasons why we haven't had a formal requirements gathering process is that we do have these astronomers on the team that are on tap that tell us the way things should be. They don't always agree with each other, but that's all part of the fun." (Interview, AstroGrid Developer, AG/10)*

> *"I think its enormously important that we do have the scientists and the tool producers working together reasonably closely on a team, in a dedicated project in a way that they can organically explore what can be done" (Interview, eMinerals Scientist, AG/2)*

Having immediate access to scientists (e.g., having instant messaging and online chatrooms so that discussions could be easily facilitated) was crucial, both to understanding requirements, and to implementing any designs. These low level detailed requirements may be obvious to a domain expert, but not to those unfamiliar with the domain. Thus, when asked about how project participants sought to understand requirements, team members often pointed out the communication channels that existed within the team. For instance, for project teams of 12 (eMinerals) and 20 (AstroGrid), there were a significant number of ways in which communication was encouraged, including email, notes on the project wiki, weekly access grid meetings, instant messaging and project mailing lists. Project participants were often quick to point out the benefits of encouraging close communications, which seemed rare in other large-scale collaborative projects:

> *"… when I was doing my PhD I was involved in a EU funded collaboration framework 4 [….] which was 6 or 7 sites across the EU looking at water in land [...]we met once a year and had a 3 day discussion meeting and that was the collaboration" (interview, eMinerals Project Scientist, EM/7)*

For the project managers of both AstroGrid and eMinerals, a significant emphasis in their work was placed on team building, and on creating a team that could function as a virtual organisation. This engagement with project scientists at close quarters enabled informal discussions in various forms concerning requirements. These informal discussions included direct questions, progress updates or presentations to facilitate knowledge transfer, and sometimes as a subtle method to encourage users to test a new version of software. The project scientists emphasised how they have had to learn to articulate their working processes and to think in much finer-grained terms than before. Here, two of the AstroGrid scientists describe their experiences of working with the developers:

> *"... I think we've all had to learn on the science side, is how to really break down the stages that well basically the heuristics of it how to break down all those little judgement calls that you make almost unconsciously and actually specify them and specify them and actually figure out what's possible and what isn't and from the developers point of view I*

think it was then making sure that they ask us the right questions ... we've had to learn what's interesting to the developers (Interview, AstroGrid Project Scientist, AG/8)

" yes sure, [laughs] its very interesting, interacting with the developers they have a completely they're used to working in a completely different way ... because they don't know the physics, obviously because they are software developers they might not always do things in the way that would be most obvious for the physicists so they might go off on a tangent while, to do a certain task, while if they talk to you about something, you might be able to point out there's another better way of doing something (Interview, AstroGrid Project Scientist, AG/5)

8.3.4. Mutual Learning of Domain knowledge

In design methodologies such as participatory design and user-centred design, the user is placed at the heart of the requirements process. It is the users needs and desires that are intended to be met in drawing up the requirements specifications. This type of relationship, where the emphasis is placed on the developer to 'collect' information from the user, is also predominant in requirements engineering. Traditional requirements elicitation techniques such as interviewing, point to the expectation that the users are the experts, while the requirements engineers (or developers, in this case) are apprentices who are learning about the domain. Rarely does the situation exist where the user is expected to understand the technical details of the software that is being developed. Through the empirical work in this thesis, one observation is that this line between users/designers/developers is much more blurred. In most cases the project scientists consider understanding the characteristics of the technology to be part of their responsibilities, which in turn helps them to understand and articulate their needs more effectively.

Let us start from the developer's perspective. Finding ways of understanding and articulating scientists' needs was not a straightforward process as observed throughout the three case studies. For the AstroGrid team, many of the developers did not have a background in astronomy. Over the course of the project, each of the developers mentioned how they picked up various knowledge and terminologies from the astronomers, and through discussing issues with the wider astronomy

community (interview, AstroGrid Developer, AG/10). As a result, these developers had to structure their group meetings and discussions very carefully to ensure that they covered the right topics. Here, the same AstroGrid developer talks about this issue of learning about the domain:

> *"… yes they [project scientists] were useful especially since so few of us were astronomers. This is a general problem again with gathering requirements. It's structuring your sessions so that there would be a bit about data access and the registry and a bit about this and the other, and its very hard when you're not familiar with the industry you're working in, to even work out how to approach requirements gathering because you're not quite sure where to start. So you start off very vague, you pick up bits, then you pick up other bits, and then you find that you've missed a whole chunk over here because nobody thought to ask and nobody thought to tell and so on, so its like that in the beginning, but it'll balance out" (Interview, AstroGrid Developer, AG/10)*

The developer above hinted at the problem of finding issues about which 'nobody thought to ask and nobody thought to tell'. This was observed in the TransProject case study during a meeting discussing the database storage structure. For several weeks the project team had talked about the database fields. The assumption by the project developer was that only single episodes of cancer had to be recorded, while the assumption by the scientists (Principal Investigators on the team) was that information about multiple episodes of cancer would be stored. When this mismatch was noticed after a prototype had been built and approved, the project team had to make significant changes to the architecture of the database, as well as other user interface elements, to make the cancer episode the central entity in the database rather than the patient.

This point was not even raised until the developer asked about another aspect of the system (i.e., whether patients should be contacted again in the future after consent). It was a situation where there was a vagueness in the discussion such that the developer did not know to ask, and the scientist did not know to mention it.

From the project scientists' perspective, they had to learn about what to explain and the level of detail to go into (as discussed in the previous section on how scientists

had to learn to articulate their thinking). In addition, there was a mutual respect for each other as this scientist explains:

> *"I think it's the combination of the two that's really the winning team, I think the physicists on their own cannot progress very far, and the software developer on their own cannot progress very far, it's the combination of the two that works" (Interview, AstroGrid Project Scientist, AG/5)*

What is perhaps unusual was that there was an onus on the project scientists to learn about the technology aspect as well. This of course varied from scientist to scientist, and project to project, but there was at least some evidence that the scientists took time to understand how Grid technologies worked as this interview extract demonstrates:

> ***Interviewer**: … and it's quite important to work together*
> ***Project Scientist**: Oh definitely, if you get on, it makes life a lot easier […]*
> ***Interviewer**: When you say that you had a lot of transfer of knowledge, do you think that's important for the developers to know about the scientists and vice versa?*
> ***Scientist**: Yeah, I know. It's necessary then if [Developer] knows how I run my calculations and what I need to look for, then the tools he develops will fit around that. And similarly I need to understand how the [Grid] tools are working so that I can use them to the optimal capacity" (Interview, eMinerals Scientist, EM/7)*

Across the cases, there was enough evidence to show there was an onus on the scientists to learn to articulate their needs more effectively by a) having a better idea of what the developers are interested in and b) using the same terminology as the developers. In summary, the scientists recognised that they needed the developers as much as the other way round.

8.3.5. No Clear 'Optimal' Team

Through the case studies, it was clear that the requirements were constructed and shaped through the close relationship that existed between the developers and project scientists. As partners within the project, they both had a substantial stake within the process. This relationship between the two groups (project scientists and the

developers), and its significance, is explored in more detailed in the previous two sections (8.2.1.2, 8.2.1.3). To an extent, this largely represents the interface between the scientific work and the technical aspects of the projects.

Membership of these two groups is clearer in some projects than others. In the AstroGrid case, there were strong distinctions between the two groups, as the project structure itself was designed to separate them, giving each clear roles. On the other hand, in the eMinerals project, such distinctions were less rigid; scientists contributed to the software, and were largely influenced by factors such as their coding abilities, or the scientific practice. The difference between these two could be explained by AG's vision of building a robust, "solidly engineered" virtual observatory. As a result, they preferred to hire software engineers with an industrial background. In addition, they had firmer ideas of e-science in astronomy and there was some recognition of these problems by certain members of the astronomy community. On the other hand, eMinerals, as a project was a lot more flexible and experimental in general. They focused on using a Grid testbed infrastructure to test various applications on how their science could be conducted using Grid technologies.

Not every project employed project scientists. The Transproject case study did not employ any project scientists, and relied instead on the Principal Investigators in charge. Two of the PIs were the initial target end-users, and were the domain experts that the project team had to engage with. Arguably, progress largely slowed due to the lack of readily available domain experts to discuss requirements with. Considering the critical role that the project scientists played over the course of the other two case studies a strong argument is provided for the inclusion of a group of dedicated end-users in the requirements process in e-science.

This discussion raises a much broader question of what expertise and groups of people are most appropriate in the development of such systems? Every team had, to a certain extent, the problem of how to deal with knowledge transfer, and the interface between understanding the science, supporting it and possibly changing methods of work inside the discipline. Longer term, such projects may lead to the

creation of new, hybrid disciplines, as has been seen for instance for combinations between science and engineering such as geo-engineering. This last point warrants a further discussion outwith this thesis; however, it is worth considering how one possible characteristic of e-science projects is that they are trying to design for hybrid disciplines that may themselves yield a further set of nascent activities. Returning to the discussion around the composition of the team, it may be that hiring researchers who have a background in both science and software engineering would aid in the efficiency and effectiveness of the project team in designing e-science tools. However, presumably such combinations of experience are rather difficult to obtain.

8.3.6. Rapidly Changing Technological Environment

> *"At the beginning you don't you're guessing, someone says go away and build a Grid. Yeah right, who's done that then? No one [laughs] So you're in the lap of the Gods, you believe it's possible but until you try, you don't know; you don't know where the pot holes are" (Interview, AstoGrid Technical Lead, AG/3)*

> *"… it became obvious that the computing problems were a show stopper. In getting the very bleeding edge middleware to do anything useful - that was really holding things up" (Interview, eMinerals Project Scientist, EM/5)*

One recurring theme emerging from the empirical work is that of the many choices related to the technical infrastructure that were tied to the external environment (i.e., outwith the project). This presents an opportunity to gain insight into what are considered as acceptable or feasible solutions for the problem at hand, through the lens of technology choice. Initial technology choice will have "strong consequences for later stages of design" (Bergman & Mark, 2002, p. 224). Technology choice has significant implications for the future of the system itself, on a number of levels. In practical terms, great consideration has to be given to the underlying technology; after all technology selection is central to understanding how requirements are satisfied and it provides the fundamental building blocks of the entire system.

In e-science, the rapidly moving technological environment within which the projects exist has been highlighted as a potential challenge in managing such projects (Buetow, 2005). During collection and analysis of the data, team members (especially those in management) spoke of the myriad of technological choices available at the start of the projects. These choices included which programming to use, the overall architecture, which implementation of a component to use, or even which (data) standards to support within the application or infrastructure. An all-encompassing vision such as e-science invites conflicting and complementary views of its implementation and realisation, and team members mentioned the numerous choices available towards the beginning of the programme. Questions arose such as: Is it better to build our own implementation of this component, or can we utilise an existing implementation? Part of this problem was the lack of existing e-science systems, or experience, to point to feasible or desired solutions. This was not helped by the fact that a number of infrastructure based projects that started around the same period needed similar generic components as each other; resulting in a number of different implementations of the same components.

8.4. Implications

8.4.1. Implications for E-Science Practitioners and Funders

8.4.1.1. Negotiations as a central aspect of Requirements Work in E-Science

One key implication in conceptualising requirements as an ongoing refinement of problems and solutions is that negotiations must be a central aspect to development work. Bergman et al. (2003) concluded that this would give rise to the 'political' requirements engineer, in their discussion of the implications arising from such a conceptualisation. They argued that leaving negotiations to senior non-technical

professionals would simply compound the problem, as they would not understand the trade-offs involved in crafting the solution. Instead it was thought that this skillset would be best incorporated into the role of the requirements engineer. Leaving aside a potentially long discussion of whether there is a place for requirements engineers in e-science (picked up in the next section on implications for RE), the bigger question posed here is whether there is a need for a technical member of staff with key negotiation skills. This staff member would be required to move between multiple levels of the project team, as well as to be responsible for some degree of project scope. For example, in the AstroGrid project, this person existed in the form of the technical manager, who was tasked to manage the technical team and craft the overall architecture of the system. For a large-scale infrastructure project this made sense, and even in smaller projects, there is merit in considering someone being responsible for the overall technical vision, with the mandate to decide what is to be built. This role already exists in other industries in various guises, most notably Directors in the Film industry.

Given the skillset that such a person would have to possess, it is questionable whether a single person could fulfil this role, or whether, such an individual could be found. If more e-science or similar projects are funded, it might be the case that the available supply of individuals could be 'grown'. However, given the assumption that this might not be the case, the lack of such individuals could pose a significant skills gap for any future e-science projects.

8.4.1.2. Separating the challenges of building e-science technologies for the first time with the challenges of on-going challenges

In reflecting on this research about the implications for e-science practitioners, it is perhaps useful to attempt to delineate between the challenges of building e-science technologies for the first time with the challenges of building e-science technologies, every time.

This research has identified multiple strands as a way of making sense of the potential reasons why requirements work in e-science proves challenging. It is important to note that the landscape in which e-science projects are now conducted is significantly different from when the research was carried out. Grid technologies and their community have inevitably grown more mature with time and use. In addition data standards now exist, where once there were none, especially in the case of first round projects like AstroGrid. These standardisations make it far easier as they stabilise parts of the solution space, and make the narrowing down to a solution much easier. These stabilised aspects of the underlying technologies and the wider data standards help provide a much more solid platform for any future endeavour.

However, shifting the focus to consider the actual work of articulating requirements, it is much less obvious whether one can draw solid implications for those involved in future e-science-like endeavours. It is fair to say that a significant difficulty in designing e-science tools in the first round of the E-Science Programme was that these projects were undertaken by project teams that had not built anything similar before. As a result, participants had to learn the process of developing these tools as well as building the solutions themselves. Many project participants noted the huge volume of learning that had to be done as part of the project: about e-science, about working in a multi-disciplinary manner, about the domain etc. This research seems to suggest that this learning aspect will never disappear and that the discomfort of having to grapple with multiple disciplines and organisations will be a persistent feature of e-science projects. Whichever lessons can be drawn from this research, they may not be able to help considerably, as such challenges are likely to be inherent to the process.

One observation is that experience counts. Previous participants in similar projects should be able to better deal with these challenges; however, the question is, have these participants learnt new skills to actually deal with these issues more efficiently? Alternatively, have they merely grown more comfortable with dealing with the uncertainties associated with such a project? In truth, it is probably a combination of

both. If that is the case, then one implication for e-science practitioners and funding bodies in general is that project teams should be funded for more than a single project, and the real benefits would begin in the second or third repeat of similar projects. Taking this idea to its limits suggests that the formation of a unit (research team) that simply specialises in developing e-science tools for each discipline could be beneficial.

8.4.1.3. Developing Robust Technologies as Software Engineering Research

In studies concerning e-science and e-infrastructure projects, attention is drawn to an uneasy tension between the project 'scientists' and the computer science researchers who are placed in the same project to develop e-science tools. On one hand, it is argued that the computer science researchers have to produce something novel in order to produce publications, while the scientists, on the other hand, would like a robust, reliable tool to be developed. While this tension was evident to an extent in each of the three case studies, another related, but over-looked aspect is that building e-science tools itself involved a lot of research by each of the project participants.

The processes of taking an idea, talking to end users and other stakeholders, and investigating the capabilities of the enabling technologies are all research tasks. Arranging all of this involves a level of uncertainty and risk typical of any research work. Yet, to some extent, existing literature seems to suggest that the development of a robust tool does not seem to be considered novel enough for computer science research.

This poses a deeper question that involves some self-reflection for computer science researchers: should this type of work, which does bear the characteristics of research work, be considered research work? And if so, where is the line drawn? Examples can be drawn to support or counter this argument. For instance, some of this work is easily identifiable as computer science research work (such as development of a new data mining algorithm in another discipline), and others are easily argued not (such

as writing a simple scheduling script). Beyond these clear cases, however, where does the boundary lie? Perhaps if software engineering existed as a discrete discipline, similar to other engineering disciplines, rather than within computer science research departments, then that would be the natural home for such research. However, because software engineering departments rarely exist separately from computer science, this debate could continue for many years to come.

8.4.2. Implications for Requirements Engineering

This research raises several questions for requirements engineering for other, non e-science, large-scale, distributed, multi-organisational systems.

8.4.2.1. Pre-empting Enabling Technologies

According to the popular saying 'when you have a hammer, everything looks like a nail'. One interesting implication resulting from seeing requirements as the navigation between problems and solutions is that solving one can limit the other. This, in principle is not a bad thing: after all, a core argument in this thesis is that those decisions are built upon over time. However, in a case such as e-science, where the enabling technology has already been 'fixed' (i.e., Grid technologies), then problems can arise because project teams will struggle to find the right problem for the solution. Both the AstroGrid and TransProject teams found out quickly that Grid technologies were not suitable for their purposes, but had to spend a lot of time discussing and convincing stakeholders higher up that they should switch to a different set of technologies (both opted for web services). The eMinerals team could afford a much more "lightweight" approach. They sought out "Grid-able problems" (section 6.5) i.e., scientific problems inside their discipline that fitted in with the Grid computing paradigm.

Thus, a strong recommendation is that in these large-scale infrastructure projects the underlying enabling technology should not be fixed at the start.

8.4.2.2. Exploration/Pre-production phase before locking down

Funding for e-science projects, like any other large-scale software development project, requires a statement of the work proposed, alongside approaches taken for the work, as well as resource estimations. This is necessary for a host of good organisational reasons, not least, to be able to provide the funders with an estimate of what to expect in return for the money provided. However, one of the dangers when work actually begins on a project that involves high levels of uncertainty (e.g., e-science tools) is that from the outset, the final solution is not clear. The technologies outlined in the proposed work are highly unlikely to be the final solution, as finding that solution is part of the work to be done. The project managers often realise this, and accept that it will be some time into the project before they know the exact scope and solution that the project team will deliver on. During this first part of the project, all three of the case studies mention a period of time where the primary function of the team was to investigate the different tools available, evaluate the different enabling technologies and to build a few prototypes, in order to clarify what could be delivered.

Looking outside the e-science and academic world, and attempting to generalise into the development of large-scale software systems, it might be useful to build this type of exploratory, examinations phase into the start of projects with similar characteristics. In creative industries such as film and animation, research and development for production is an accepted part of their process, and is termed 'pre-production'. Traditionally, this does not exist in software development. There is no clear explanation for this, apart from the possibility that such a phase is perhaps unnecessary in a commercial context. While it might not be applicable to all systems, it is certainly worthwhile considering the possibility of a pre-production phase for

large-scale systems that stretch across multiple organisations, and involve five or more years of development time. The pre-production phase would be a research and development phase, with a small team of people to do some initial investigation of the problem and solution spaces. The potential benefits of such an approach would far outweigh the cost of failure of the entire project.

8.4.2.3. Unresolved discussion – Does E-science need a Requirements Engineer?

Finally, to close the implications section, I discuss whether e-science projects (and other similar systems) need a requirements engineer.

Part of the challenge of answering this question is (at the risk of stating the obvious): *what does a successful requirements engineering phase of an e-science project look like?* Is it a well-written specification document? Conversely, is it the production of a widely adopted tool built on e-science technologies? If it is the former, then project participants will argue that the longitudinal nature of e-science systems make specifications defined at the start of a six year project redundant, especially due to the fast moving nature of Information Technologies. If it is the latter, then it is well established that there are a much wider set of factors at play in successful technology adoption (Voss et al., 2007).

For any large-scale system, where the central argument is that requirements are built up over time, having a single point where the problem and the solutions are decided can be problematic, in the sense of getting a single individual (or team) to approve scope for the whole project. They would have to be responsible for all aspects of the build. As mentioned above, for such an engineer to exist, their skillset would need to incorporate user needs, technical needs, negotiation skills and business skills. Lloyd and Simpson (2005) suggest that the requirements process for e-science might be more akin to product, rather than systems development due to the lack of a clear user or deliverable. They discuss the need to identify the market/need, the difficulty of drawing up a set of agreed baseline requirements, and the production of technical

requirements from non-uniform working practices. If this is true, then maybe it will not be a requirements engineer that will be required in future similar builds, but rather, a software product manager.

Regardless of what the engineer might be called, it is clear from this research that articulating requirements in these systems is incredibly complex. As a result, there is huge value in having someone on the team who appreciates requirements techniques and can understand how to work with users and other stakeholders in order to help the team's developer navigate through this complex landscape.

8.4.3. Implications for Future Research

This research has thrown up several questions in need of further investigation. These questions could have implications for future research within the fields of e-science and requirements engineering.

This thesis' contribution is to provide a strategic overview of the sorts of challenges that exist when articulating requirements in e-science projects and for e-science technologies. Over the course of the research, it became rapidly obvious that e-science projects exist in different sizes, scopes and purposes; some projects aim to deliver an e-infrastructure for an entire discipline, whilst others are one-off, self-contained projects. While a number of studies have been carried out to date (including this one) on requirements and design work in e-science projects, the diversity of such projects - at times - make it challenging to grasp exactly which insights from lessons learnt are applicable to which project; i.e., how can we best guide practitioners about to embark on similar software development projects? One possible tool for navigating practitioners and researchers alike is to create a framework that maps different dimensions of e-science projects. For instance, dimensions could include size of project, team diversity, aim/purpose of project and maturity of software used. These dimensions would be mapped onto expected problems, such that similar problems are not rediscovered over and again, but can be

anticipated so that appropriate resources, skills and timescales can be drafted into play. Projects and their studies can be mapped onto this framework to enable future practitioners to grapple with which studies (and insights) would provide the most relevance.

Another emerged implication is the question – to what extent are the tasks being undertaken during e-science development more akin to research, rather than traditional software development. Research projects contain an element of 'risk' that the outcome will not work. In this context, the software being developed might not be feasible because the project could involve attempts at novelty on a number of fronts; as a result, the uncertainty and ambiguity involved in building a 'successful' tool often multiplies. For practitioners, it is often daunting and challenging to navigate the first steps of such a project. It could be fruitful to create frameworks that encourage participants to reflect upon the risks and ambiguities involved and suggest ways to estimate them. Recommendations could be made as a result. For instance, the framework could include the consideration of building a user interface toolkit to aid front-end software development for projects that are concerned with user engagement, creation of a portfolio, balancing novel and 'tried and tested' technologies, or aligning with technological trends if possible.

Temporality has become an emergent area of research inside the e-infrastructure literature (Karasti et al., 2010; Ribes and Finholt, 2009). In this thesis, the consideration of temporal aspects of development work was evident throughout the chapters, especially in the AstroGrid case study. In that case, the study was built around the existence of multiple temporal elements in the development e-infrastructure, e.g., monthly rhythms, phases and lifelong technology trajectories. In addition, it was highlighted how some of these rhythms can clash. These observations are similar to those of other scholars such as Karasti et al. (2010) who describe the tensions between balancing the short-term demands within a project, with the longer-term considerations for the construction of infrastructure. Thus, a potential useful avenue of research is to extend this discussion, and explore the

strategies employed in balancing the multiple demands across different rhythms, then consider whether these strategies can be formalised. The benefits of this research would be to inform practitioners of strategies and tactics they could employ during e-science projects, or whilst designing and developing similar software.

8.5. Reflections on Research Design

To recap, the research methodology was to:

- Select three case studies to understand some of the difficulties attached to working out requirements for e-science technologies;
- Select three projects which varied in size, domain and progression stage;
- Use a combination of semi-structured interviews and observation to collect data.

At the end of the process, in reflecting on the research design, it becomes rapidly apparent that the design of the study was very ambitious. At the start of the project, there were strong reasons for selecting three cases that differed.

Research around requirements engineering, software engineering and project management in general surrounding e-science projects were few at the start of the research period (of course, this has been rectified now with a number of studies cited in this thesis). However, at the beginning of the research, of those papers that did emerge at that time, most were based on a number of similar projects all in the same domain (Biology). As a result, this thesis aimed to detail a number of studies from outside that domain. The decision was then made to select three projects from different domains, rather than a single domain, in order to build depth into the research and insights produced. This was done in an attempt to increase the generalisability of the results across the empirical work. Undoubtedly, this choice

made the data analysis phase unruly and complex. For each case study, the project structure had to be understood, and information on the progression of the projects, the nuances of the disciplines that they sat within, and the multiple organisations that they worked within had to be gathered too. Each case study (of which, there were three) became significantly challenging to analyse alone, but was even more so, when attempting to draw cross case comparisons.

Looking back, a significant driver for this aim of "generalizability" came from the first case study: TransProject. One research issue that is sometimes mentioned but never emphasised enough is the extent to which conflict and contention arises inside a single e-science project. By their very nature, e-science projects have to work across multiple organisations and multiple disciplines. Issues are flagged up, and sometimes partners fall out. It is never completely clear whether insights generated from understanding conflicts inside a single project, are only applicable to that project, or whether they are applicable to e-science projects in general. Yet, sometimes these conflicts of opinion affect the approaches that project participants take. The tension between what is generalisable and what is not, is a factor researchers have to contend with on a regular basis; however, the specific/general boundary is much more challenging to navigate because of the scale of this type of cross-disciplinary large scale software development, compared to for instance, a well-understood process like building a website.

As a result as the case studies were rather diverse (in aim, scope, size and domain), the sorts of conclusions that could be reached by the study are, to a certain extent, sometimes quite general and at others, fairly specific. At times, the diversity played a helpful part in shedding light on whether an observation in a study was a one-off, or something worth investigating because it occurred across all three cases. At other times it was a hindrance, especially in relation to the complexity of attempting to pull together all three relatively diverse approaches. Attempting to make sense of three very diverse studies, and how all the different strands came together, was not straightforward, and in fact took a significant commitment of time. It can be argued

that findings are certainly more at a general level: they can apply to all e-science projects, not just ones of a specific domain or size. However, great care has gone into the discussion and implications sections to ensure that as much as possible, the study offers insights that cut across all three case studies. I believe this is a key advantage in a comparative study like this.

8.6. Conclusions

The fieldwork in this thesis points out the challenging but important role played by requirements work that permeates e-science software projects. In the complex and provisional settings of these emerging e-science platforms, requirements engineering was shown to play an important role in setting and communicating the direction of the project. Uncertainty surrounding the direction of the project means requirements are subject to repeated conflicts and revisions. The teams using the requirements engineering tools cooperate but that cooperation arises from a process of questioning, disputing and revising the requirements that are shared across groups and locations.

In e-science projects we encounter a high level of diversity in terms of the cross-disciplinary teams working on developing applications and infrastructures combined with constraints in terms of the initial infrastructures and the modes of working. In such a complex setting, striving for complete agreement is almost invariably a futile pursuit. In attempting to develop software to meet a collection of potentially mutually unsatisfiable needs, requirements engineering plays an important role as a focus for the resolution of issues between groups of collaborators. Requirements contribute to making progress on projects undertaken by diverse, interdependent, groups working to create software that meets some or all needs that are imperfectly captured by the requirements. The three case studies considered in this thesis illustrate how requirements are subject to continual negotiation and re-shaping throughout the lifetime of the e-science project. The case studies are heterogeneous

in terms of scale and domain and each illustrates different strategies for working to stabilize requirements and make them useful to most of the project groups. AstroGrid illustrates how disputes around requirements reflect the temporal rhythm of the project with peaks of disagreement around transitions from phase to phase in the project where debate is essential to secure a route to make progress. E-Minerals illustrates the prototyping approach that is adoptable by smaller projects where the requirements are captured through experimentation using prototypes. This strategy is certainly appropriate for smaller projects but it does pose problems in terms of the mobility or generalizability of the project experience. TransProject illustrates the issues that arise when the "solution set" is underspecified and an important part of the requirements process is an attempt to stabilise just what constitutes the infrastructure upon which to build.

This thesis contributes to the literature by providing a strategic overview of the negotiating moves undertaken by various groups and how these are played out as challenges involved in stabilising requirements to help direct engineering work in e-science projects. The detailed observational studies combined with this strategic overview will be of interest to researchers studying Requirements Engineering, researchers studying the structure and evolution of large-scale projects, and practitioners interested in undertaking projects tackling the challenges of developing infrastructure and applications to support science based on complex data infrastructures.

Bibliography

Arskey, H., & Knight, P. (1999). *Interviewing for Social Scientists.* London: Sage.

AstroGrid. (2003). *AstroGrid 2 Project Proposal.*

AstroGrid. (2003). *AstroGrid Phase A Report.*

AstroGrid. (2001). *AstroGrid Project Proposal.*

Atkins, D. (2003). *Revolutionizing Science and Engineering Through Cyberinfrastructure.* Report of the National Science Foundation Blue-Ribbon Panel on Cyberinfrastructure.

Atkinson, M., Crowcroft, J., Goble, C., Gurd, J., Rodden, T., Shadbolt, N., et al. (2002). *Computer Challenges to emerge from e-Science.* Retrieved from National E-Science Centre: http://umbriel.dcs.gla.ac.uk/nesc/general/news/uk_escience_agenda.html

Ballejos, L., & Montagna, J. (2008). Method for stakeholder identification in interorganizational environments. *Requirements Engineering , 13* (4), 281-297.

BBSRC. (2004). *BBSRC Bioinformatics and e-ScienceGrant Holder's Workshop Report.*

Beckles, B. (2005). Re-factoring grid computing for usability. In *Proceedings of the UK e-Science All Hands Meeting.* pp. 75-82.

Beckles, B., Brostoff, S., & Ballard, S. (2004). A first attempt: initial steps toward determining scientific users' requirements and appropriate security paradigms for

computational grids. *In Proceedings of the Workshop on Requirements Capture for Collaboration in e-Science.* pp. 17-43.

Benbasat, I., Goldstein, D., & Mead, M. (1987). The Case Study Research Strategy in Studies of Information Systems. *MIS Quarterly , 11* (3), 369-386.

Bergman, M., & Mark, G. (2002). Technology Choice as a First Step in Design: The Interplay of Procedural and Sensemaking Processes. *Proceedings of the 4th conference on Designing interactive systems: processes, practices, methods, and techniques*, (pp. 224-234). London.

Bergman, M., King, J. L., & Lyytinen, K. (2002a). Large Scale Requirements Analysis Heterogeneous Engineering. *Scandinavian Journal of Information Systems* , 37-55.

Bergman, M., King, J. L., & Lyytinen, K. (2002b). Large-Scale Requriements Analysis Revisited: The need for understanding the Political Ecology of Requirements Engineering. *Requirements Engineering* , 152-171.

Berman, F., & Hey, T. (2004). The Scientific Imperative. In I. Foster, & C. Kesselman, *The Grid2: Blueprint for a New Computing Infrastructure* (pp. 13-24). San Francisco: Morgan-Kaufman Publishers.

Berman, F., Fox, G., & Hey, T. (2003). The Grid: past, present, future. In F. Berman, G. Fox, & T. Hey, *Grid Computing: Making the Global Infrastructure a Reality.*

Boehm, B. (2002). Get Ready for Agile Methods, with Care. *Computer* , pp. 64-69.

Boehm, B. W. (1988, May). A Spiral Model of Software Development and Enhancement. *Computer* , 61-72.

Boehm, B. W. (1981). *Software engineering economics.* Prentice-Hall.

Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., et al. (2004, Feb). *W3C Working Group Note 11 February 2004.* Retrieved from Web Services Architecture: http://www.w3.org/TR/ws-arch/

Bowker, G. C., Star, S.L. (1999). *Sorting Things Out: Classification and Its Consequences*. MIT Press: Cambridge.

Brekke, J. S., & K. Palinkas, L. A. (2007). Translational Science at the National Institute of Mental Health: Can Social Work Take Its Rightful Place? *Research on Social Work Practice* , 123-133.

Brooks, F. (1975). *Mythical Man-Month, The: Essays on Software Engineering.* Addison-Wesley.

Bubenko Jr, J. A. (1995). Challenges in Requirements Engineering. *Proceedings of the Second IEEE International Symposium on Requirements Engineering* , 160-162.

Burgess, R. G. (1984). *In the Field: An Introduction to Field Research.* London and New York: Routledge.

Button, G., & Sharrock, W. (1994). Occasioned Practices in the Work of Software Engineers. In M. Jirtoka, & J. Goguen, *Requirements Engineering: Social and Technical Issues* (pp. 217-240).

Button, G., & Sharrock, W. (1996). Project Work: The Organisation of Collaborative Design and Development in Software Engineering. *Computer Supported Cooperative Work* , 369-386.

Button, G., & Sharrock, W. (1998). The Organizational Accountability of Technological Work. *Social Studies of Science , 28* (1), 73-102.

Calleja, M., Bruin, R., Tucker, M., Dove, Tyer, R., Blanshard, L., et al. (2005). Collaborative grid infrastructure for molecular simulations: The eMinerals minigrid as a prototype integrated compute and data grid. *Molecular Simulation* , 303-313.

Cambrosio, A., Keating, P., Mercier, S., Lewison, G., & Mogoutov, A. (2006). Mapping the emergence and development of translational cancer research. *European Journal of Cancer* , 3140-3148.

Carusi, A., & Jirotka, M. (2006). *Building Virtual Research Environments and User Engagements.*

Celis, J. E., & Gromov, P. (2003). Proteomics in translational cancer research: Toward an integrated approach. *Cancer Cell* , 9-16.

Chin, J., & Coveney, P. (2004). *Towards tractable toolkits for the Grid: a plea for lightweight, usable middleware.* UK E-Science Technical Report Series.

Coughlan, J., & Macredie, R. D. (2002). Effective Communication in Requirements Elicitation: A Comparision of Methodologies. *Requirements Engineering , 7*, 47-60.

D'Agostino, G., Hinds, C., Jirotka, M., Meyer, C., Piper, T., Rahman, M., et al. (2008). On the importance of intellectual property rights for e-science and the integrated health record. *Health Informatics* , 95-111.

D'Adderio, L. (2004). *Inside the Virtual Product.* Cheltenham: Edward Elgar.

David, P., & Spence, M. (2003). *Towards Institutional Infrastructures for e-Science: The Scope of the Challenge.*

Davidson, E. J. (2002). Technology Frames and Framing: A Socio-Cognitive Investigation of Requirements Determination. *MIS Quarterly , 26* (4), 329-358.

Davis, A. M. (1993). *Software Requirements: Objects, Functions and States.* NJ: Prentice Hall International.

Davis, A., & Hickey, A. (2002). Requirements researchers: Do we practice what we preach? *Requirements Engineering Journal* , 107-111.

Davis, G. B. (1982). Strategies for Information Requirements Determination. *IBM Systems Journal , 21* (1), 4-30.

de la Flor, G., Lloyd, S., Jirotka, M., & Warr, A. (2007). Designing Software in Support of Workplace Activities - Embedding E-Science Applications. *Third International Cofnerence on e-Social Science. Michigan.*

de Roure, D., Jennings, N. R., & Shadbolt, N. (2003). The Semantic Grid: A future e-Science infrastructure. In F. Berman, G. Fox, & T. Hey, *Grid Computing - Making the Global Infrastructure a Reality* (pp. 437-470). John Wiley and Sons Ltd.

Dittrich, Y. (2002). Doing Empirical Research on Software Development: Finding a Path between Understanding, Intervention, and Method Development. In *Social Thinking - Software Practice* (pp. 243-262). MIT Press.

Djorgovski, S. G. (2005). Virtual Astronomy, Information Technology, and the New Scientific Methodology. *CAMP '05: Proceedings of the Seventh International Workshop on Computer Architecture for Machine Perception* (pp. 125-132). Washington: IEEE Computer Society.

Djorgovski, S. G., & Williams, R. (2005). Virtual Observatory: From Concept to Implementation. *ASPCS , 345*, 517.

Dove, M. T., Calleja, M., Bruin, R., Wakelin, J., Tucker, M. G., Lewis, G. J., et al. (2005a). The eMinerals collaboratory: Tools and Experience. *Molecular Simulation* , 329-337.

Dove, M., White, T., Bruin, R., Tucker, M., Calleja, M., Artacho, E., et al. (2005b). eScience usability: the eMinerals experience. *Proceedings of E-Science All Hands Meeting 2005*, (pp. 30-37).

Dove, M., Artacho, E., White, T., Bruin, R., Tucker, M., Murray-Rust, P., et al. (2005c). The eMinerals project: developing the concept of the virtual organisation to support collaborative work on molecular-scale environmental simulations. *Proceedings of the UK e-Science All Hands Meeting*, (pp. 1058-1065).

Dowson, M. (1987). Iteration in the Software Process. *Proceedings of the 9th International Conference on Software Engineering.*

Eberlein, A., & Leite, J. C. (2002). Agile Requirements Definition: A View from Requirements Engineering. *Proceedings of the International Workshop on Time-Constrained Requirements Engineering.*

Edwards, P., Jackson, S., Bowker, G., & Knobel, C. (2007). *Understanding Infrastructure: Dynamics, Tensions and Design.*

Eisenhardt, K. M. (1989). Building Theories from Case Study Research. *The Academy of Management Review , 14* (4), 532-550.

eMinerals. (2002). *eMinerals 1 Proposal.*

eMinerals. (2005). *eMinerals 2 Proposal.*

EPSRC. (2004). *e-Science 2004: The Working Grid.*

e-Science Core Programme structure and key activities. (n.d.). Retrieved from Research Councils UK: http://www.rcuk.ac.uk/escience/coreprog/default.htm

Finholt, T., & Birnholtz, J. (2006). If we build it, will they come? The cultural challenges of cyberinfrastructure. In W. Bainbridge, & M. Roco, *Managing nano-bio-info-cogno innovations* (pp. 89-101). Springer Netherlands.

Finkelstein, A. (1994). Requriements Engineering: a review and research agenda. *IEEE Software Engineering Conference*, (pp. 10-18).

Foster, I. (2002). The Grid: A New Infrastructure for 21st Century Science. *American Institute of Physics , 55*, 42-47.

Foster, I., Kesselman, C., & Tuecke, S. (2001). The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing , 15* (3), 200-222.

Foster, I., Kesselman, C., Nick, J., & Tuecke, S. (2002). The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. *Open Grid Service Infrastructure WG, Global Grid Forum.*

Freeman, P. A. (2007). Is 'designing' cyberinfrastructure -or even, definining it-possible? *First Monday* .

Fry, J., & Thelwall, M. (2006). Using Domain Analysis and Organisational Theory to Understand E-Science Sustainability. *Second International Conference on e-Social Science.* Manchester.

Fujimura, J. H. (1987). Constructing 'Do-able' Problems in Cancer Research: Articulating Alignment. *Social Studies of Science* , 257-293.

Galison, P. L., & Hevly, B. W. (1992). *Big Science: The Growth of Large-Scale Research.* Stanford University Press.

Galison, P., & Hevly, B. (1992). *Big Science: The Growth of Large-Scale Research.* Stanford: Stanford University Press.

Gavaghan, D., Whiteley, J., Pitt-Francis, J., Slaymaker, M., Lloyd, S., David Boyd, M., et al. (2004). Gathering requirements on an Integrative Biology project.

Gentzsch, W. (2006). National Grid Initiatives: Lessons Learned and Recommendations. *Second International Conference on e-Science and Grid Technologies* (p. 6). Amsterdam: IEEE Computer Society.

Gerring, J. (2007). *Case Study Research: Principles and Practices.* Cambridge: Cambridge University Press.

Gibson, A., Stevens, R., Cooke, R., Brostoff, S., & schraefel, m. c. (2006). *myTea: Connecting the Web to Digital Science on the Desktop.*

Gillham, B. (2000). *Case Sudy Research Methods.* London, New York: Continum.

Goguen, J., & Linde, C. (1993). Techniques for Requirements Elicitiaton. In S. Fickas, & A. Finkelstein (Ed.), *Requirements Engineering* (pp. 152-164). IEEE Computer Society.

Goguen, J. A. (1996). Formality and Informality in Requirements Engineering. *Proceedings of the 2nd international Conference on Requirements Engineering (ICRE '96)* . Washington: ICRE. IEEE Computer Society.

Grudin, J. (1990). The computer reaches out: the historical continuity of interface design. In J. C. Chew, & J. Whiteside (Ed.), *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Empowering People* (pp. 261-268). Seattle, Washington, United States: ACM.

Hartswood, M., Ho, K., Procter, R., Slack, R., & Voss, A. (2005). Etiquettes of Data Sharing in Healthcare and Healthcare Research. *Proceedings of the 1st International Conference on e-Social Science.* Manchester.

Hartswood, M., Jirotka, M., Procter, R., Slack, R., Voss, A., & Lloyd, S. (2005). Working IT out in e-Science: Experiences of requirements capture in a HealthGrid project. *Studies in health technology and informatics*, *112(1)*, 198-209.

Hertzum, M. (2004). Small-Scale Classification Schemes: A Field Study of Requirements Engineering. *Computer Supported Cooperative Work , 13* (1), 35-61.

Hey, T., & Trefethen, A. (2003). The Data Deluge: An e-Science Perspective. In G. Berman, G. Fox, & T. Hey, *Grid Computing – Making the Global Infrastructure aReality* (pp. 809-824). Chichester: Wiley.

Hey, T., & Trefethen, A. (2002). The UK e-Science Core Programme and the Grid. *Future Generation Computer Systems* , 1017-1031.

Hickey, A. M., & Davis, A. M. (2003). Elicitation Technique Selection: How Do Experts Do It? *RE '03: Proceedings of the 11th IEEE International Conference on Requirements Engineering* (p. 169). Washington: IEEE Computer Society.

Hine, C. (2005). Material culture and the shaping of e-science. *First International Conference on e-Social Science.* Manchester.

Jacobson, I., Christerson, M., Jonsson, P., & Overgaard, G. (1993). *Object-Oriented Software Engineering.* Addison-Wesley.

Jirotka, M., Procter, R., Hartswood, M., Slack, R., Simpson, A., Coopmans, C., et al. (2005). Collaboration and Trust in Healthcare Innovation: The eDiaMoND Case Study. *Computer Supported Cooperative Work* , 369-398.

Jordan, K., & Lynch, M. (1993). The Mainstreaming of a Molecular Biological Tool. In G. Button, *Technology in Working Order: Studies of Work, Interaction and Technology* (pp. 160-180). London: Routledge.

Karasti, H., Baker, K. S., & Millerand, F. (2010). Infrastructure time: long-term matters in collaborative development. *Computer Supported Cooperative Work (CSCW)*, *19*(3-4), 377-415.

Knorr-Cetina, K. D. (1999). *Epistemic Cultures. How the Sciences Make Knowledge.* Cambridge: Harvard University Press.

Kotonya, G., & Somerville, I. (1998). *Requirements Engineering.* Chichester, New York: John Wiley and Sons.

Krasner, H. (1989). Requirements Dynamics in Large Software Projects. *11th World Computer Congress.* Amsterdam, Netherlands.

Latour, B. (1987). *Science in Action: How to Follow Scientists and Engineers through Society.* Harvard University Press.

Lawerence, K. A. (2006). Walking the Tightrope: The Balancing Acts of a Large e-Research Project. *Computer Supported Cooperative Work , 15* (4), 385-411.

Lee, C. P., & Bietz, M. (2009). Barriers to Adoption of New Collaboration Technologies for Scientists. *CHI 2009.* Boston: ACM.

Lee, C., Dourish, P., & Mark, G. (2006). The Human Infrastructure of Cyberinfrastructure. *Computer Supported Cooperative Work* (pp. 483-492). New York: ACM Press.

Lewis, G., Hasan, S., Alexandrov, V., Dove, M., & Calleja, M. (2005). Multicast application sharing tool - Facilitating the eMinerals virtual organisation. *Lecture Notes in Computer Science 3516* , pp. 359-366.

Lloyd, S., & Simpson, A. (2005). Project Management in Multi-Disciplinary Collaborative Research. *International Professional Communication*, (pp. 602-610).

Lloyd, W. J., Rosson, M. B., & Arthur, J. D. (2002). Effectiveness of Elicitation Techniques in Distributed Requirements Engineering. *Requirements Engineering, IEEE International Conference on* (p. 311). Los Alamitos: IEEE Computer Society.

Lubars, M. D., Meredith, G., Potts, C., & Richter, C. (1992). Object-Oriented Analysis for Evolving Systems. *ICSE*, (pp. 173-185).

Luff, P., Hindmarsh, J., & Heath, C. (2000). *Workplace Studies: Recovering Work Practice and Informing System Design.* Cambridge: Cambridge University Press.

Lynch, M. (1997). *Scientific practice and ordinary action : ethnomethodology and social studies of science.* Cambridge: Cambridge University Press.

Macaulay, L. A. (1996). *Requirements Engineering.* London: Springer.

Maiden, N. A. (1998). CREWS-SAVRE: Scenarios for Acquiring and Validating Requirements. *Automated Software Eng. , 5* (4), 419-446.

Maiden, N. A., & Rugg, G. (1996). ACRE: Selecting methods for Requirements Acquistion. *Software Engineering Journal , 11* (3), 183-192.

Mascord, M. (2006, October 13). The Integrative Biology VRE Project Presentation. Oxford.

McCray, P. W. (2000). Large Telescopes and the Moral Economy of Recent Astronomy. *Social Studies of Science* (5), 685-711.

McLaughlin, J., Rosen, P., Skinner, D., & Webster, A. (1999). *Valuing Technology: Organisations, Culture and Change.* London: Routledge.

Miles, S., Wong, S. C., Fang, W., Groth, P., Zauner, K.-P., & Moreau, L. (2007). Provenance-based validation of e-science experiments. *Web Semantics* , 28-38.

Momtahan, L., & Martin, A. (2002). e-Science experiences: Software Engineering practice and the EU DataGrid. *Proceedings Asia-Pacific Software Engineering Conference* (pp. 269-275). IEEE Press.

Monteiro, E., & Svanæs, D. (1993). The role of empirical evidence in software engineering. *Proc. NIK '93.* Tapir.

Nuseibeh, B., & Easterbrook, S. (2000). Requirements Engineering: A Roadmap. *Proceedings of the Conference on the Future of Software Engineering* (pp. 35-46). Limerick, Ireland: ACM.

Pardridge, W. M. (2003). Translational science: what is it and why is it so important? *Drug Discovery Today* , pp. 813-815.

Pollock, N., Williams, R., & D'Adderio, L. (2007). Global Software and its Provenance: Generification Work in the Production of Organizational Software Packages. *Social Studies of Science* , 254-280.

Potts, C., Takahashi, K., & Antón, A. (1994). Inquiry-Based Requirements Analysis. *IEEE Software , 11* (2), 21-32.

Procter, R., Hartswood, M., Voss, A., Slack, R., Rouncefield, M., & Buscher, M. (2009). Concluding Remarks. In A. Voss, M. Hartswood, R. Procter, M. Rouncefield, R. Slack, & M. Buscher, *Configuring User-Designer Relations: Interdisciplinary Perspectives* (pp. 219-229). London: Springer.

Reddy, M., & Dourish, P. (2002). A finger on the pulse: temporal rhythms and information seeking in medical work. *Proceedings of the 2002 ACM conference on Computer Supported Cooperative Work* (pp. 344-353). New Orleans: ACM.

Reddy, M., & Dourish, P. (2002). A finger on the pulse: temporal rhythms and information seeking in medical work. In E. F. Churchill, J. McCarthy, C. Neuwirth,

& T. Rodden (Ed.), *Proceedings of the 2002 ACM conference on Computer supported cooperative work*, (pp. 344-353). New Orleans, Louisiana, USA.

Ribes, D., & Finholt, T. (2009). The Long Now of Technology Infrastructure: Articulating Tensions in Development. *Journal of the Association for Information Systems* , 375-398.

Robson, C. (1993). *Real World Research.*

Robson, C. (2002). *Real World Research* (2nd Edition ed.). Oxford: Blackwell.

Roy, A. E., & Clarke, D. (2003). *Astronomy: Principles and Practice.* Institute of Physics Publishing.

Royce, W. (1970). Managing the Development of Large Software Systems.

schraefel, m. c., & Dix, A. (2007). Within Bounds and Between Domains: Reflecting on Making Tea within the Context of Design Elicitation Methods. *International Journal of Human Computer Studies* .

schraefel, m. c., Hughes, G., Mills, H., Smith, G., Payne, T., & Frey, J. (2004). Breaking the Book: Translating the Chemistry Lab Book into a Pervasive Computing Lab Environment. *CHI.* Vienna.

Silverman, D. (2002). *Doing Qualitative Research: A Practical Handbook.* London: Sage.

Simmhan, Y. L., Plale, B., & Gannon, D. (2005). A survey of data provenance in e-science. *SIGMOD Rec.* , 31-36.

Sommerville, I. (2006). *Software Engineering* (8th Edition ed.). Harlow: Addison Wesley.

Stevens, P., & Pooley, R. (1999). *Using UML: Software Engineering with Objects and Components.* Harlow: Pearson.

Suchman, L. (1987). *Plans and situated actions : the problem of human-machine communication.* New York: Cambridge University Press.

Sutcliffe, A. (1996). A Conceptual Framework for Requirements Engineering. *Requirements Engineering , 1*, 170-189.

Swartout, W., & Balzer, R. (1982). On the inevitable intertwining of specification and implementation. *Communication of the ACM , 25* (7), 438-440.

Synder, C. (2003). *Paper Prototyping.* Morgan Kaufmann.

Szalay, A. S., & Brunner, R. J. (1999). Astronomical archives of the future: a virtual observatory. *Future Gener. Comput. Syst. , 16* (1), 63-72.

Szalay, A., & Gray, J. (2006). The World Wide Telescope. *MSDN Magazine , 21.5*.

Turner, W., Bowker, G., Gasser, L. and Zacklad, M. (2006). 'Information Infrastructures for Distributed Collective Practices', *Computer Supported Cooperative Work*, 5(1), pp. 93 – 110.

Tyer, R., Blanshard, L., Dam, K. K., Allan, R., Richards, A., & Dove, M. (2003). Building the eMinerals Minigrid. *Proc UK e-Science All Hands Meeting 2003.* EPSRC.

Tyer, R., Kleese van Dam, K., Couch, P., Todorov, I., Bruin, R., Walker, A., et al. (2006). Automatic Metadata Capture and Grid Computing. *Proceedings of the UK e-Science All Hands Meeting* (pp. 381-384). NeSC.

Ure, J., Procter, R., YuWei, L., Hartswood, M., Anderson, S., Lloyd, S., et al. (2009a). A Socio-technical Perspective on the Evolution of eHealth Infrastructure. *Journal of the Association for Information Systems* .

Ure, J., Rakebrandt, F., Lloyd, S., Khanban, A., Procter, R., Anderson, S., et al. (2009b). Giving Them Something to Hate: Using Prototypes as a Vehicle for Early Engagement in Virtual Organizations. *Social Science Computer Review* .

van Lamsweerde, A. (2009). *Requirements Engineering.* John Wiley & Sons.

van Lamsweerde, A. (2000). Requirements Engneering in the Year 00: A Research Perspective. *International Conference on Software Engineering*, (pp. 5-21).

Vessey, I., & Conger, S. A. (1994). Requirements specification: learning object, process, and data methodologies. *Communications of the ACM* , 102-113.

Villers, S., Bowers, J., & Rodden, T. (1999). Human Factors in Requirements Engineering: A survey of human sciences literature relevant to the improvement of dependable systems development processes. *Interacting with Computers , 11* (6), 665-698.

Voss, A. (2009). *e-Uptake Final Report.* JISC.

Voss, A., Procter, R., Jirotka, M., & Rodden, T. (2007). Managing User-Designer Relationships in e-Research Projects. *Workshop on Realising e-Research Endeavours.* Edinburgh.

Walsh, J. P., & Bayma, T. (1996). Computer Networks and Scientific Work. *Social Studies of Science , 26* (3), 661-703.

Walton, N. (2004). Meeting the User Science Challenge for a Virtual Universe. *Toward an international virtual observatory: proceedings of the ESO/ESA/NASA conference* (pp. 187-192). Garching, Germany: Springer.

Walton, N., Lawerence, A., & Linde, A. E. (2004). AstroGrid: powering science from multistreamed data. *Proceedings of SPIE - The International Soceity for Optical Engineering* (pp. 146-152). International Society for Optical Engineering.

Walton, N., Lawerence, A., & Linde, A. (2003). Scoping the UK's Virtual Observatory: AstroGrid's Key Science Drivers. *Astronomical Data Analysis Software and Systems XII ASP Conference Series*, (pp. 25-28).

Warr, A., de la Flor, G., Jirotka, M., & Lloyd, S. (2007). Usability in e-Science: The eDiaMoND Case Study. *CHI.*

Webb, C. P., & Pass, H. I. (2004). Translation research: from accurate diagnosis to appropriate treatment. *Journal of translational medicine* .

Wehling, M. (2006). Translational medicine: can it really facilitate the transition of research "from bench to bedside"? *European Journal of Clinical Pharmacology* , 91-95.

Welsh, E., Jirotka, M., & Gavaghan, D. (2006). Post-genomic science: cross-disciplinary and large-scale collaborative research and its organizational and technological challenges for the scientific research process. *Philiosophical Transactions of The Royal Society* , 1533-1549.

Williams, R., & Edge, D. (1996). The social shaping of technology. *Research Policy , 25* (6), 865-900.

Woolgar, S. (1994). Rethinking requirements analysis: Some implications of recent research into producer-consumer relationships in IT development. In M. Jirtoka, & J. Goguen, *Requirements Engineering: social and technical issues.*

Wouters, P., & Beaulieu, A. (2006). Imagining E-Science Beyond Computation. In C. M. Hine, *New Infrastructures for Knowledge Production: Understanding E-Science* (pp. 48-70). Hershey: Information Science Publishing.

Yin, R. K. (2002). *Case Study Research, Design and Methods* (3rd ed.). Newbury Park: Sage Publications.

Yin, R. K. (1994). *Case Study Research: Design and Methods* (2nd Edition ed.). London: Sage.

Zimmerman, A. (2007). A socio-technical framework for cyberinfrastructure design. *e-Social Science Conference.* Ann Arbor.

Zimmerman, A., & Faniel, I. M. (2006). Stakeholders at a Distance: User Participation in the Development of Cyberinfrastructure. *Computer Supported Cooperative Work (CSCW 2006).*

Zimmerman, A., & Nardi, B. A. (2006). Whither or Whether HCI: Requirements Analysis for Multi-Sited, Multi-User Cyberinfrastructures. *CHI* (pp. 1601-1606). Montreal: AM Press.