Acceptance of Domain-Specific Visual Languages for Environmental Scientific Computing

Luke Nguyen-Hoan

February 10, 2014

A thesis submitted for the degree of Doctor of Philosophy of the Australian National University
Declaration

I declare that this thesis is my own, and to the best of my knowledge it does not contain any materials previously published or written by another person except where otherwise stated.

Luke Nguyen-Hoan
Acknowledgements

Firstly, I would like to thank my supervisors, Shayne Flint, Ramesh Sankaranarayana, and Geoff Cary, for their support and guidance throughout my candidature. Their experience and constructive feedback has been invaluable throughout the research presented in this thesis.

In addition to my supervisors, I would like to thank Lynette Johns-Boast for her support and re-energising conversations. I also thank Ian Davies for his insightful comments regarding some of his experiences as a scientific software developer, as well as his help with LAMOS throughout.

I would like to thank the many people who kindly donated their time and effort to participate in the surveys and studies of this research, as without their participation none of this research would have been possible.

I also would like to thank many of my fellow PhD candidates, who have been sounding boards and sanity checks for ideas throughout my research.

Finally, I would like to thank my parents, who have supported (and tolerated) me throughout the long process of completing this thesis.
Summary

The lack of software engineering in scientific software development has previously been thought to be the reason for many of the issues and problems faced in that area. This thesis presents one manner in which one particular software engineering methodology, domain-specific visual languages, could be of greater utility and more accepted in the scientific software area. This application of domain-specific visual languages is justified through a survey of current scientific software development practices. Studies of the acceptability of tools for specifying environmental software simulations are performed. These studies compare an existing form based tool, a domain-specific visual language tool, and finally an extended domain-specific visual language tool which allows users to record context and provenance information alongside the experimental setup. The results show that this last use of domain-specific visual language techniques is positively received by potential users.

Publications


  - The survey described in this paper is covered in greater detail in Chapter 2.
Abstract

Software is developed and used for many different scientific purposes. However, the development of scientific software has traditionally involved little application of software engineering. This thesis describes an investigation of the use of software engineering in scientific software development, culminating in a study of the application of an existing software engineering methodology, namely domain-specific visual languages, to the area of scientific software. This research focuses on the acceptability of the domain-specific visual language approach by environmental scientists, resulting in a new manner of recording context and provenance information using the domain-specific visual language paradigm, which integrates such information alongside experimental specifications and designs.

The application of domain-specific visual languages to the scientific software field was justified based on the results of an initial survey, which investigated the current state of scientific software development. In particular, the use of software engineering techniques, the lack of which was thought to be the cause of many issues in scientific software development, was investigated. The results identified that although the situation had improved from that reported in previous literature, there remained areas for potential improvement. The proven benefits of domain-specific visual languages, demonstrated in other areas of software use, matched up well with the areas of potential improvement in scientific software.

Although the benefits of domain-specific visual languages has been studied in detail, there has been little attention paid to the acceptance of the approach by users, particularly in the scientific software area. Therefore this research focused on the acceptability of the domain-specific visual language approach by users in the scientific software area. A study was performed comparing the acceptance of a domain-specific visual language based tool and an existing form based tool, both of which are used to specify an environmental sciences experiment as a software simulation. The two tools were found to not differ significantly in acceptance, although the domain-specific visual language tool was perceived to be significantly
better for understanding, exploring, and communicating the experiment itself.

Additional information collected in the study revealed that environmental scientists focused on context information, which explains what is being tested and why it is of interest, when explaining an experimental scenario. Apart from its commonplace inclusion in scientific papers, there has been little research into how context information is recorded by scientists, although there exist some tools for the related area of provenance information recording. A second study was performed to measure the acceptance of an extended domain-specific visual language tool, which allowed scientists to record context and provenance information in an integrated environment alongside the experimental setup.

This new method of recording context and provenance information, integrated with experimental setup within a domain-specific visual language, received positive feedback from potential users. Potential benefits of this approach are that it forces experiment developers to explicitly consider and capture context and provenance information while creating their experiments, and allows them to directly link such information with components of the experimental design.
# Contents

Acknowledgements \hspace{20pt} v  
Summary \hspace{20pt} vii  
Abstract \hspace{20pt} ix  
Table of Contents \hspace{20pt} xi  
List of Figures \hspace{20pt} xvii  
List of Tables \hspace{20pt} xix  
Table of Abbreviations \hspace{20pt} xxi  

## 1 Overview

1.1 Introduction \hspace{20pt} 1  
1.2 Motivation \hspace{20pt} 1  
1.3 Context \hspace{20pt} 2  
	h.3.1 Why scientific software development is different \hspace{20pt} 2  
	h.3.2 Software engineering and scientific software \hspace{20pt} 4  
1.4 Research aim \hspace{20pt} 6  
1.5 Thesis scope \hspace{20pt} 7  
1.6 Thesis structure \hspace{20pt} 7  

## 2 Scientists Developing Software

2.1 Introduction \hspace{20pt} 13  
2.2 Existing literature \hspace{20pt} 14  
	2.2.1 Subsequent literature \hspace{20pt} 16  
2.3 Goals and hypotheses \hspace{20pt} 16  
2.4 Methodology \hspace{20pt} 18
2.4.1 Survey conduct ........................................................................ 18
2.4.2 Participants ........................................................................... 18
2.4.3 Survey contents ................................................................... 19

2.5 Results and discussion ................................................................ 20
2.5.1 Demographics ...................................................................... 20
2.5.2 Programming languages ...................................................... 23
2.5.3 Development tools ............................................................... 26
2.5.4 Development teams and user base characteristics ................. 27
2.5.5 Documentation .................................................................... 28
2.5.6 Testing and verification ........................................................ 32
2.5.7 User and stakeholder confidence ........................................... 35
2.5.8 Non-functional requirements ................................................ 35
2.5.9 Threats to validity ............................................................... 42
2.5.10 Comparison with previous research ...................................... 43
2.5.11 Summary of results compared to hypotheses ......................... 45

2.6 Implications .............................................................................. 47

2.7 Subsequent literature .................................................................. 48
2.7.1 Summary of subsequent literature ........................................... 49
2.7.2 Comparison of results with subsequent research ...................... 50

2.8 Conclusion ............................................................................... 50

3 Scientists and DSVLs ..................................................................... 53
3.1 Introduction ............................................................................... 53
3.2 Existing literature ....................................................................... 55
3.2.1 Benefits of DSLs and DSVLs ................................................ 55
3.2.2 Acceptance of DSLs and DSVLs ............................................ 56
3.2.3 DSLs and DSVLs for scientific modeling ................................. 57

3.3 Goals and hypotheses ................................................................ 57

3.4 Methodology ............................................................................. 58
3.4.1 Experimental conduct ......................................................... 58
3.4.2 Participants .......................................................................... 59
3.4.3 Sessions and activities ........................................................ 60
3.4.4 Tools .................................................................................... 63
3.4.5 Statistical analysis approach ............................................... 67

3.5 Results and discussion .............................................................. 67
3.5.1 Diagrams ............................................................................. 68
3.5.2 Acceptance of DSVL ............................................................ 73
## CONTENTS

3.5.3 Threats to validity .................................................. 79
3.5.4 Summary of results compared to hypotheses ............... 80
3.6 Implications ................................................................. 81
3.7 Conclusion ................................................................. 82

4 Scientists, Software, and Systems .................................. 85
   4.1 Introduction ............................................................. 85
   4.2 Background .............................................................. 85
      4.2.1 Systems .......................................................... 86
      4.2.2 Software systems .............................................. 87
      4.2.3 Systems engineering .......................................... 88
      4.2.4 Software engineering ......................................... 89
      4.2.5 Science .......................................................... 89
   4.3 From a systems thinking point of view ......................... 92
      4.3.1 Thinking about scientific research ....................... 92
      4.3.2 Thinking about software engineering ...................... 93
      4.3.3 Differences between scientific research and software engineering ........................................... 94
   4.4 A systems thinking approach to scientific software .......... 95
   4.5 DSVLs and context for scientific experiments ............... 97
   4.6 Conclusion ............................................................. 98

5 DSVLs, Scientists, Context, and Provenance ..................... 101
   5.1 Introduction ............................................................. 101
   5.2 Existing literature .................................................. 103
   5.3 Goals and hypothesis ............................................... 105
   5.4 Methodology ........................................................... 106
      5.4.1 Experimental conduct ....................................... 106
      5.4.2 Participants .................................................. 107
      5.4.3 Sessions and activities ..................................... 108
      5.4.4 Extended DSVL tool ......................................... 110
      5.4.5 Statistical analysis approach .............................. 110
   5.5 Results and discussion ............................................ 112
      5.5.1 Qualitative comments - current practices for context and provenance ........................................... 112
      5.5.2 Acceptance of extended DSVL tool ....................... 116
5.5.3 Qualitative comments - context and provenance in the extended DSVL tool ........................................ 121
5.5.4 Threats to validity ...................................................... 122
5.5.5 Summary of results compared to hypotheses .................. 123
5.6 Implications .............................................................. 123
5.6.1 Current practices for context and provenance ......... 123
5.6.2 Acceptance of the extended DSVL tool ................. 125
5.7 Conclusion .............................................................. 128

6 Conclusion ................................................................. 131
6.1 Summary ................................................................. 131
6.2 Contribution ............................................................ 131
6.3 Limitations ............................................................... 132
6.4 Implications ............................................................. 133
  6.4.1 Problems with scientists' acceptance of DSVLs .... 133
  6.4.2 Thinking about software engineering within systems . 136
  6.4.3 Environmental scientists recording context and provenance using DSVLs ........................................ 137
  6.4.4 A lack of context recording by scientists ............... 138
  6.4.5 A lack of systematic provenance recording by scientists .... 139
6.5 Future work ............................................................... 139
  6.5.1 Context and provenance recording ......................... 140
  6.5.2 Acceptability ....................................................... 140
  6.5.3 Developing DSVLs from domain expert diagrams ........ 141
  6.5.4 Scientific software developers and DSVLs ............... 142
  6.5.5 Other ways in which software engineering can help .... 142
6.6 Overall conclusion .................................................... 143

A Survey: Ethics Approval .................................................. 145
B Survey: Scientific Software Development ......................... 153
C DSVL Study: Ethics Approval ......................................... 163
D DSVL Study: Information Sheet ...................................... 171
E DSVL Study: Consent Form ............................................. 175
F Environmental Scenario Description ................................ 179
CONTENTS

G  DSVL Study: Diagrams By Participants 181
H  DSVL Study: Questionnaire 193
I  Extended DSVL Study: Information Sheet 203
J  Extended DSVL Study: Consent Form 207
K  Extended DSVL Study: Questionnaire 211

Bibliography 217
List of Figures

1.1 Thesis structure and flow of ideas ........................................ 8
2.1 Respondent location by country ........................................... 21
2.2 Type of organisation worked for ......................................... 22
2.3 Highest level of education completed ................................... 22
2.4 Level and field of education .............................................. 23
2.5 Programming experience ................................................... 24
2.6 Programming languages used .............................................. 25
2.7 Reasons for Choice of Programming Language ....................... 25
2.8 Types of tools used .......................................................... 26
2.9 Reasons why tools are used ................................................ 28
2.10 Development team sizes .................................................... 29
2.11 Intended user base sizes .................................................... 29
2.12 Programming experience of intended user base ...................... 30
2.13 Types of documentation commonly produced ....................... 31
2.14 Reasons for and against production of documentation .......... 32
2.15 Testing and verification activities commonly performed .......... 33
2.16 Reasons for performing testing and verification .................... 34
2.17 Methods used to increase user and stakeholder confidence ...... 36
2.18 Importance of non-functional requirements as rated by respondents 38
2.19 How often respondents developed for particular user base sizes, respondents categorised by usability rating .................. 41
2.20 How often respondents developed for users with, without and both with and without programming experience, respondents categorised by usability rating ........................................... 42
3.1 Baseline tool (LAMOS) ....................................................... 65
3.2 DSVL tool ................................................................. 66
3.3 Part of a diagram drawn by a participant .............................. 69
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4</td>
<td>Example decision tree graph drawn by a participant</td>
<td>70</td>
</tr>
<tr>
<td>3.5</td>
<td>Qualities of diagrams drawn</td>
<td>71</td>
</tr>
<tr>
<td>3.6</td>
<td>Information in diagrams drawn</td>
<td>72</td>
</tr>
<tr>
<td>3.7</td>
<td>Perceived usefulness of DSVL tool</td>
<td>73</td>
</tr>
<tr>
<td>3.8</td>
<td>Perceived usefulness of Baseline tool</td>
<td>74</td>
</tr>
<tr>
<td>3.9</td>
<td>Perceived ease of use of DSVL tool</td>
<td>75</td>
</tr>
<tr>
<td>3.10</td>
<td>Perceived ease of use of Baseline tool</td>
<td>75</td>
</tr>
<tr>
<td>3.11</td>
<td>Perceived modeling utility of DSVL tool</td>
<td>77</td>
</tr>
<tr>
<td>3.12</td>
<td>Perceived modeling utility of Baseline tool</td>
<td>77</td>
</tr>
<tr>
<td>4.1</td>
<td>The system of scientific research</td>
<td>91</td>
</tr>
<tr>
<td>4.2</td>
<td>Influence diagram of scientific research activities (in the area of scientific computing) and software engineering tools, techniques, and methodologies</td>
<td>96</td>
</tr>
<tr>
<td>5.1</td>
<td>Modeling Experience</td>
<td>107</td>
</tr>
<tr>
<td>5.2</td>
<td>Extended DSVL tool</td>
<td>111</td>
</tr>
<tr>
<td>5.3</td>
<td>Qualitative Comments regarding Context and Provenance</td>
<td>113</td>
</tr>
<tr>
<td>5.4</td>
<td>Perceived usefulness of Extended DSVL tool</td>
<td>116</td>
</tr>
<tr>
<td>5.5</td>
<td>Perceived ease of use of Extended DSVL tool</td>
<td>118</td>
</tr>
<tr>
<td>5.6</td>
<td>Perceived modeling utility of Extended DSVL tool</td>
<td>120</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Combined *important* and *very important* ratings for non-functional requirements ........................................ 38
2.2 Survey results compared to past papers ........................................ 44
3.1 Perceived usefulness - difference between DSVL and Baseline tools 74
3.2 Perceived ease of use - difference between DSVL and Baseline tools 76
3.3 Perceived modeling utility - difference between DSVL and Baseline tools ........................................ 78
5.1 Perceived usefulness - difference between Extended DSVL and Original DSVL tools ........................................ 117
5.2 Perceived usefulness - difference between Extended DSVL and Baseline tools ........................................ 117
5.3 Perceived ease of use - difference between Extended DSVL and Original DSVL tools ........................................ 118
5.4 Perceived ease of use - difference between Extended DSVL and Baseline tools ........................................ 119
5.5 Perceived modeling utility - difference between Extended DSVL and Original DSVL tools ........................................ 120
5.6 Perceived modeling utility - difference between Extended DSVL and Baseline tools ........................................ 121
# Table of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANU</td>
<td>Australian National University</td>
</tr>
<tr>
<td>DSL</td>
<td>Domain-Specific Language</td>
</tr>
<tr>
<td>DSVL</td>
<td>Domain-Specific Visual Language</td>
</tr>
<tr>
<td>GEMS</td>
<td>Generic Eclipse Modeling System[124]</td>
</tr>
<tr>
<td>GPL</td>
<td>General Purpose Language</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language[102]</td>
</tr>
<tr>
<td>UTAUT</td>
<td>Unified Theory of Acceptance and Use of Technology[122]</td>
</tr>
<tr>
<td>TAM</td>
<td>Technology Acceptance Model[24, 25]</td>
</tr>
<tr>
<td>TAM2</td>
<td>Extended Technology Acceptance Model[121]</td>
</tr>
</tbody>
</table>
Chapter 1

Overview

1.1 Introduction

This thesis describes exploratory research of software engineering in scientific areas, leading to the investigation of the acceptability of Domain-Specific Visual Languages (DSVLs) by scientists and the effect of tailoring DSVLs for scientists.

This chapter provides an overview of the research, first with the motivation for conducting the research presented in this thesis. Second, the background context of the research is described. Third, the research aims are listed. Fourth, the scope of this thesis is defined. Finally, the structure and contents of this thesis will be summarised.

1.2 Motivation

The initial motivation of this research was to determine how software engineering could be of aid to scientists. This motivation was based upon previous literature, detailed in section 1.3, which indicated that one of the significant issues facing scientific software development was the lack of use of software engineering practices.

The overarching aim of this research is to improve the scientific productivity of scientists, through improvements in scientific software development.

From this initial motivation and based on the literature summarised in section 1.3, preliminary research was conducted into the area of scientific development. This preliminary research, which is reported in Chapter 2, investigated the current usage of software engineering practices in scientific software development. The results of this research indicated that basic software engineering practices are
more widely used in scientific software development than previously expected. The previously held notion of a lack of software engineering in scientific software development was no longer such a significant issue, although some gaps still remain.

Following the results of the preliminary research, it was determined that DSLs could be of use in the scientific software community, and more specifically computational environmental science. There exists evidence regarding the usefulness of Domain-Specific Languages (DSLs) and DSVLs in non-scientific areas. Many of the general benefits of DSLs and DSVLs would also be useful in the scientific domain. Additionally, there have been DSLs and DSVLs developed for use in computational environmental science. Thus far research has focussed on producing new DSVLs and DSLs with little analysis of the acceptability of such approaches in the scientific software area. From this, the following research questions were posed:

1. Are DSVLs accepted by scientists?

2. Why are DSVLs accepted or not accepted by scientists?

3. How can the acceptability of DSVLs by scientists be increased?

1.3 Context

This section explores why scientific software in particular is of interest to the software engineering community. The factors which differentiate scientific software development from other areas of software development will be explored, as well as previously proposed and trialled improvements to scientific software development.

The development of software for scientific purposes has been paid increasing attention by the software research community, for example in the special issue of IEEE Software [35] on developing scientific software, and the International Workshop on Software Engineering for Computational Science and Engineering [15].

1.3.1 Why scientific software development is different

As described by Segal and Morris. [113] ‘developing scientific software is fundamentally different from developing commercial software’. There are two main reasons for this difference: that non-software specialists are often the developers
for the software, and that there is often incomplete understanding of the domain which leads to an inability to specify requirements up-front and difficulties in validating the resulting software. Each of these reasons will be explored in turn.

**Scientific software developers**

Scientific software is often developed by non-software specialists, or 'professional end user developers' as denoted by Segal [111]. Professional end user developers are 'people such as research scientists who work in highly technical, knowledge-rich domains and who develop software in order to further their professional goals' and 'have no formal training in software engineering'. In comparison, software professionals have formal training in software development, and do not develop software in order to further their own goals as professional end user developers do, but instead develop software in accordance with requirements provided by (or elicited from) other people such as scientists.

Professional end user developers pose a challenge in the improvement of scientific software development due to their lack of formal training, instead having gained their knowledge of programming in the early stages of their research careers through 'prototype or throw-away code' [90]. To exacerbate this issue, Segal [111] notes that often professional end user developers were students or short-term researchers, and their involvement in software development ended as they gained permanent positions. This lack of focus and support for software development within their domains leads to an unstable community. In addition, professional end user developers are more dependent upon co-located communities, unlike software engineers who have a wide network of fellow practitioners to draw knowledge from.

The primary focus of scientists - including professional end user developers - is to perform science, not develop code. Therefore, when looking to improve the way in which scientific software is developed, improving the way in which scientists perform science must be the primary goal, and care must be taken to not be sidetracked into improving scientific software merely for the sake of improving scientific software. As described by Basili [7] in the case of scientists developing for high performance computing systems, the focus of scientific researchers is on producing publishable results and maximising scientific output. However, Segal [112] indicates that this focus on the scientific outcomes presents a barrier to the adoption of software engineering practices, due to the low importance that professional end user developers place on software development knowledge and
skills. Section 1.3.2 presents the asserted benefits of applying software engineering practices to the area of scientific software development.

The interaction between the aforementioned professional end user developers and software professionals, as well as scientists who do not perform development themselves but are users of the software, has also received limited attention. Segal [112] explores the clashes which occur when software engineers and scientists attempt to impose their software development practices upon each other. In particular, differing expectations in requirements collection and specification were identified in a number of case studies [110, 111].

**Incomplete domain understanding**

A common purpose of scientific software is to improve domain or scientific understanding. Incomplete domain understanding often translates to exploration and experimentation in science. However, due to this incomplete domain understanding, when software for scientific purposes is first created, there is often only a vague idea of what is required, and the requirements of the software being developed are not fully known. This incomplete understanding of the domain has two main effects: firstly that requirements are unable to be fully specified up-front, and secondly that verification is difficult [113].

As the requirements of the software are unable to be fully specified up-front [113], an iterative development process is often adopted, where code is developed, reflected upon, and then redeveloped, taking into account the new knowledge gained [112]. Sanders and Kelly [105] determined that this cycle allows developers to update both the code and the underlying theory as their understanding improves. In cases where the theory is well understood, the iterative approach allows the gradual introduction of more features. This iterative software development cycle reflects the iterative nature of scientific research [104].

Incomplete domain understanding also leads to a lack of known correct data to compare results with [7, 71, 105, 111]. This problem is particularly apparent in simulation software, where 'the science is too complex, too large, too small, too dangerous, or too expensive to explore in the real world' [113].

**1.3.2 Software engineering and scientific software**

The analysis of how scientific software is currently developed has resulted in a number of proposed improvements. Some of these have been trialled, with varying degrees of success.
Many of the improvements focus on the transfer of software engineering practices to the scientific software development community. Some recommend a general approach of tailoring existing practices, trialling them, and then spreading those practices which are successfully adopted by scientific software developers [7]. Others advocate that scientific software developers should adopt specific practices and specific types of development tools [10, 57, 111]. A common theme is that tools should require minimal changes to existing work practices [29, 111]. Other general advice is that tools should support the iterative development cycle which is common in scientific software development [111], be incremental in development and provide immediate return for time and effort [29].

As identified previously in Section 1.3.1, differing expectations in requirements is one of the problems faced by the intersection of software engineering and scientific software development. Although imposing up-front design and determining verification methods ahead of time has been proposed [10], other solutions may be more acceptable to scientists such as recording requirements via test cases, as advocated by Heroux and Willenburg [57] and demonstrated by Macaulay et al. [83].

The nature of the documentation for scientific software is also explored, with differing opinions. Baxter proposes sufficient documentation for anyone to be able to easily read and follow the software [10], whereas Heroux and Willenburg proposes barely-sufficient, source-centric documentation [57]. Wilson et al. [126] advises documentation of the design and purpose of the software, rather than the mechanics. Generation of source code documentation has successfully been used [80], however issues with document storage have also been encountered [2].

A more formalised approach towards testing and verification is also recommended [10, 57, 111], and successful use of regression and integration testing has been shown by Easterbrook et al. [31]. The use of issue-tracking tools has been identified as a best practice by a number of sources [10, 57, 126]. Issue-tracking software has been successfully used by Ackroyd et al. [2], although some problems with testing and continuous integration still exist in that project.

The use of other tools to support development activities is also recommended. Version control is commonly advocated [10, 57, 126] and successfully used [2, 85]. Build tools have also been advocated [57, 126] and successfully used [2, 80, 85, 106].

Modeling and computer-aided software engineering (CASE) tools has received mixed success. Unified modeling has successfully been used by Easterbrook et al. [31]. Li [80] found that requirements modeling was found to improve commu-
nication between stakeholders. However, Li also found existing CASE tools to require a significant amount of software engineering knowledge, which presented a barrier to scientific software developers.

Software lifecycle and development methodologies have successfully been used in a number of instances in scientific software development. Practices similar to those of agile software development have found acceptance in scientific software projects [31, 106]. Some extreme programming (XP) [11] practices, such as small releases, simple design, and coding standards, have been successfully used [2], although other XP practices such as pair programming and collective code ownership have only been partially been implemented in that project. User-centred design, where development and software use is driven by user testing and collaborative design workshops between scientists and developers, has successfully been applied [83].

Finally, there exist scientific workflow management systems such as Chimera [40], Kepler [4], PASOA [88], Taverna [62], and Velo [46]. These systems aim to improve the productivity of scientists by allow them to build, automate, and record their scientific computing workflows, including data management. The sharing and distribution of these workflows has also received some attention, with the myExperiment Virtual Research Environment [30] aiming to enable the sharing of workflows via techniques adopted from online social networking and Web 2.0 [95].

Further literature regarding current practices and potential areas of improvement in scientific software development is summarised in Chapter 2, as a prelude to the survey of current scientific software development practices described in that chapter.

1.4 Research aim

The overall aim of the research presented in this thesis is to investigate the use and acceptance of software engineering in the field of environmental scientific computing.

This aim will be advanced by:

- Investigating the current use of software engineering practices in scientific software development, and identify potential areas of improvement (Chapter 2).
1.5. **THESIS SCOPE**

- Investigating one such potential area of improvement in greater detail, and identify how software engineering can be of use in this area (Chapter 3).

- Identifying and testing ways in which the previously identified application of software engineering to the field of scientific software development can be improved (Chapters 4 and 5).

### 1.5 Thesis scope

The scope of this thesis is the investigation of scientific software development, the acceptance of DSVLs by computational environmental scientists, and the analysis of one method for improving the acceptability of DSVLs by computational environmental scientists. The broader implications and applicability of this work to areas other than environmental scientific computing are also discussed. However, experimentation to evaluate the broader implications and applicability of this work is outside of the scope of this thesis.

### 1.6 Thesis structure

Each chapter of this thesis covers one aspect of the overall research presented. The diagram presented in Figure 1.1 depicts the structure and flow of ideas in this thesis.

Note that the discussion of related work to this thesis is spread across Chapters 2, 3, and 5, as this thesis covers a number of different but related areas within the broad field of scientific computing. Instead of presenting all of the literature related to the entirety of the thesis in a single chapter or section, the relevant literature is instead described in the individual chapters where the literature is directly related to the work presented.

Chapters 2, 3, and 5 each present a study of one specific aspect of scientific software. Each of these chapters presents a review of the most relevant literature to the study, along with the aims, hypotheses, methodology, and results of the study. A brief discussion is also included in each chapter regarding the results of each individual study.

Chapter 4 presents a discussion regarding the broader implications of the combined results from the previously presented studies.

The flow of ideas and results throughout this thesis is described in greater detail below, with each chapter explored in turn.
CHAPTER 1. OVERVIEW

Chapter 1
Overview

[Motivation to improve scientific productivity by improving scientific software development]

Chapter 2
Scientists Developing Software

Survey of scientific software developers

[The idea to use DSVLs to improve scientific software]

Chapter 3
Scientists and DSVLs

Experiment: Acceptance of a DSVL by scientists

Chapter 4
Scientists, Software, and Systems

Experiment: Using a DSVL to record context and provenance

Chapter 5
DSVLs, Scientists, Context, and Provenance

Chapter 6
Conclusion

Plans for future research

Figure 1.1: Thesis structure and flow of ideas
Chapter 1 - Introduction

This is the current chapter, which presents the motivation and aim for the research reported in this thesis. This chapter also presents relevant background literature which explores why scientific software is of particular interest.

Chapter 2 - Scientists Developing Software

This chapter presents initial research into the practices currently used in the development of scientific software. This investigation was conducted for three reasons:

1. To gain greater understanding of the current situation of scientific software development
2. To identify any changes following previously reported research
3. To identify areas where software engineering could be of benefit

As mentioned previously, the most relevant literature to the study is also presented within this chapter. In this case, this literature covers the current practices used in scientific software development, along with the identification of possible areas of improvement.

The results of this study shows that many of the issues considered to be affecting scientific software development are not as significant as previously thought. Although there remains room for further improvement, these improvements indicate that some basic software engineering practices have been successfully adopted by the scientific software development community. Therefore, the focus of software engineering in scientific software development can be turned to the introduction, development, and adaptation of other software engineering methodologies and techniques which could improve scientific software development.

Areas in which further improvements are possible are tool use, documentation, testing, and verification. Other results of interest include that the most common reason for using tools is to improve ease of coding, and that the most important non-functional requirements for scientific software developers are reliability and functionality, closely followed by usability, maintainability, and availability.

This study resulted in the identification of DSVLs as an appropriate technology to investigate for a number of reasons. The benefits of DSVLs have
been shown to mirror those of DSLs [70], and these benefits include improved validation [8, 14], documentation [56, 77], reliability [56, 73], and maintainability [120, 73, 86]. All of these areas of improvement would be of particular benefit to the field of scientific software, as identified in this study.

Chapter 3 - Scientists and DSVLs

Although DSVLs would be of benefit in the scientific software field, these benefits are of no value if users do not accept DSVLs. This chapter describes a study of the acceptance of DSVLs by environmental scientists. This study not only aimed to determine whether DSVLs would be accepted by environmental scientists for the purpose of environmental modeling, but also aimed to determine what factors affect this acceptability.

Included in this chapter is a summary of recent literature, which covers the benefits of DSVLs as well as current uses of DSVLs and DSLs in scientific areas.

In addition to the benefits of DSVLs described previously, DSVLs also improve productivity [120, 56, 73, 77, 86] and reduce the amount of programming knowledge required by users [86]. The intellectual effort required between conceptualisation and realisation is also reduced [76]. All of these would be of additional benefit to scientists, by reducing the time required to produce scientific software and therefore increasing the time and resources available for scientific research.

Environmental modeling was chosen as the area of focus for this study for several reasons. A clearly defined subset of scientists was chosen in order to narrow the focus of the study, rather than attempting to cover the wide variety of scientific areas and uses of software for scientific purposes. Environmental modeling was chosen as this subset as DSLs have previously been developed for use in this field. Modeling is used in many scientific areas and is a significant component of scientific research [43].

The study described in this chapter compares the acceptance of two tools used to specify environmental simulations. These two tools are an existing non-DSVL tool, the LAndscape MOdeling Shell (LAMOS) developed by Lavorel et al. [78], which was used as a baseline, and a DSVL tool which was developed specifically for this research, implemented within the Generic Eclipse Modeling System (GEMS) [124] on the Eclipse IDE [33]. Both of
these tools are further described in Section 3.4.4.

The results show that although the DSVL tool was not significantly more accepted by environmental scientists as compared to the baseline non-DSVL tool for environmental modeling, the DSVL was significantly better for understanding, exploring, and communicating an experiment. Additional information collected from participants, in the form of drawings to explain an environmental scenario, indicate that environmental scientists focus on the context information of an experiment, specifically the parameters of interest, input data, and expected results.

Current DSLs and DSVLs in the area of environmental science concentrate on the specification of experimental scenarios [12, 36, 91]. The lack of support for recording context and provenance information was identified as a possible improvement to the DSVL tool.

Chapter 4 - Scientists, Software, and Systems
This chapter presents a wider view from a systems thinking perspective of the research presented thus far in the thesis, in order to identify and justify a new application of software engineering within scientific research. Brief explanations of the concepts of systems, software systems, systems engineering, software engineering, and science are provided. The systems thinking approach is applied to both scientific research and software engineering, and the differences in focus compared and contrasted. The system of scientific research is defined, and used to identify existing roles which software and software engineering play within this system. Finally, new roles which could be played by software and software engineering within the system of scientific research are explored, resulting in the identification of one such role which warrants further investigation based on the results of the study presented in Chapter 3: recording of context information, supported by the use of DSVLs.

Chapter 5 - DSVLs, Scientists, Context, and Provenance
Based on the results of the study presented in Chapter 3, the DSVL tool was expanded to allow users to also record context and provenance information alongside the specification of an environmental software simulation. This chapter details a study of the acceptability of this extended DSVL approach. In order to provide background information, the current practices of environmental scientists regarding the treatment and recording of
context and provenance information was also investigated. Details regarding the extended DSVL tool, which was purpose-built for this study, are included in Section 5.4.4.

Similar to the previous chapters, this chapter also includes a summary of the most relevant literature for this study. Although context information is commonly presented in scientific papers, there has been little research regarding how this information is managed outside of publications. However, the related area of provenance information has been researched and existing methods of recording such information is explored.

Environmental modeling was once again used as the area of focus for this study. This was to enable comparison of the results from this study with those from the previous study presented in Chapter 3. The reasons provided for this choice for the previous study also apply, namely to narrow the focus to a specific subset of scientists, and environmental modeling already having some exposure to DSLs.

The results of this study show that environmental scientists are willing to use a DSVL for recording context and provenance information. However, there was no significant difference in acceptability between the DSVL which allowed users to record context and provenance information, and the DSVL which did not.

It was also determined that the current practices of computational environmental scientists rarely involve the systematic or formal recording of context and provenance information. Software support for the recording of either type of information was limited to databases of dataset metadata.

Chapter 6 - Conclusion

The final chapter summarises the work presented in this thesis. This chapter also describes the main contributions of this research and discusses its limitations. Possible future work and potential implications are also presented.
Chapter 2

Scientists Developing Software

2.1 Introduction

The people involved in the development of software for scientific research purposes differ from those normally considered by mainstream software engineering. Domain specialists - scientists - tend to lack formal information technology or software engineering education, yet are involved in the development of software regardless of this lack of knowledge[109]. The development of software for scientific research purposes has been previously asserted to suffer from a lack of basic software engineering tools, techniques, and methodologies.

This chapter presents a survey of scientific software developers, which was conducted in order to achieve two main research goals. The first research goal was to improve understanding of the current state of development of software for scientific research purposes, and in particular investigate more broadly the lack of software engineering in the development of software for scientific research purposes which is implied by existing literature. The second research goal was to identify areas where software engineering can reasonably be applied to the development of software for scientific research purposes, in order to improve the manner in which such software is produced.

First, this chapter provides an overview of the literature regarding the development of software for scientific research purposes, which was published at the time this survey was performed. Second, the purpose and goals of this survey are detailed. Third, the experimental setup is described. Fourth, the results of the survey are presented and discussed, including a comparison of the results with those in subsequent literature. Finally, the implications of this survey are explored.
2.2 Existing literature

The development of software for scientific purposes received increased attention in the software engineering research community in the period of 2007-2009. Studies have identified particular problem areas in scientific software development through the use of case studies.

As a task in isolation, software development is not considered important by scientists. Instead, the focus of scientists on performing scientific research, and development of software is seen as just another supporting activity [7, 68, 109, 111]. This focus on scientific research, rather than software development, leads to software attributes such as performance being important but only to a limited extent [72]. In extreme cases, software development within scientific communities is not considered to be 'real work' [111]. This difference in focus impacts upon many areas of scientific software development.

Software development methodologies which find greater acceptance by scientific software developers are those which more closely match the incremental, cyclic nature of scientific research [72]. Incremental development and content delivery find greater acceptance amongst scientists [29]. Agile approaches and extreme programming (XP) [11] have been successfully used [2]. In other studies, most of the development teams utilised some kind of agile or iterative process [16, 104, 105, 111], although it was not always recognised by developers that the development process was of an agile or iterative nature [16].

Scientific software tends to be developed using lower-level programming languages. Programming language choice is driven by how easy languages are to learn, stability, and how confident developers are in the translation from programming code to the resulting machine instructions [16]. Performance and portability are considered, with the importance of these aspects varying from secondary [16] to critical [37]. Legacy code and long project lifespans mean that older programming languages still being in use [72, 104, 105]. Due to this experience, programming languages such as Fortran 77 and C++ which have a long and well-known history of use are still chosen for new projects [37, 104, 105]. However, there has been some experience with using higher-level programming languages for prototyping or visual interface support [105].

The use of tools to support scientific software development varies between projects. Some studies report little use of tools, and especially Integrated Development Environments (IDEs) and frameworks finding little acceptance due to lack of trust in their functions and what the tool is doing [7, 16, 127]. A lack of
use of version control has also been identified [127]. However, other projects indicate widespread use of tools [2, 85]. A common theme across studies regardless of the level of tool use identified is a scepticism of new technologies, primarily due to unclear benefits of adoption [7, 16, 127], and uncertainty regarding their future longevity [7]. Successful adoption of tools has been attributed to support from both users and management, strong development communities for the tools, shallow learning curves which allow users to see immediate benefits, effort made to support migration, and ongoing in-house support [85].

Documentation of scientific software programs is rarely produced. Documentation of programming code tends to be lacking, with greater emphasis placed on the scientific theories underpinning the software rather than the software itself [31, 105]. The production of documentation has been mostly driven by outside forces such as management, and there is a perception that documentation in general is useless [109], with asking a developer directly regarding the software being viewed as more effective than using documents [111]. Documentation has been shown to be insufficient for constructing and communicating understanding between software engineers and research scientists [110].

Testing and verification is considered difficult in scientific software for a number of reasons. There is often a lack of previously known correct data to compare results against [7, 16, 71]. Even in cases where such data exists finding the source of any problems is difficult as there are many possible sources of error, ranging from the scientific theory itself, the mathematics of the theory, through to the implementation of the theory in the software code [16, 68]. Further aspects of scientific software which confound testing and verification are the software's complexity and complex dynamic interactions, as well as an inability of scientists to view the software as a separate entity which requires specific attention compared to the science behind the software. [68]. Finally, time pressures and not specifically setting aside time and effort result in testing and verification often being neglected [105, 110, 111]. All of these factors result in a general lack of testing and verification of scientific software [68, 90, 104, 105, 110, 111].

The software qualities which are considered important in scientific software are particularly interesting considering all of the previously explored practices which are used or not used in scientific software development. Although the software itself is considered of secondary importance to scientific research, the performance of scientific software has been identified as an important quality [7, 16, 37, 72]. Perhaps as a result of the lack of testing and verification, as well as considerations of future use and applicability, maintenance is also identified as
an important quality [7, 16]. Similarly for reasons of longevity, portability is also considered important [7, 16, 37]. However, in line with the focus of scientists being on scientific research, and supported by the use of lower-level languages due to developer confidence and little use of tools due to lack of confidence, correctness [16, 68, 104], reliability [29], and traceability [104] are also considered important by scientific software developers.

2.2.1 Subsequent literature

Literature which was published after the study presented in this chapter is summarised and explored in Section 2.7.

2.3 Goals and hypotheses

A common theme throughout the research literature regarding scientific software development is the limited use of software engineering methodologies and techniques. However, much of the literature only relates experiences from a single project or from a small number of case studies. The wider applicability of the conclusions presented in the research literature are unknown. This survey was designed to gain some insight into the current overall status of scientific software development and how widespread the challenges and issues identified in the existing research are.

This survey has two overall goals:

1. Understand the current situation of software engineering in scientific software development
2. Identify areas where software engineering could benefit scientific software development

In order to address these goals, the following research questions and hypotheses were posed:

1. What programming languages are used and why?

**Hypothesis 2.1.** The most common programming languages in scientific software development are lower-level programming languages, such as FORTRAN and C. Higher-level languages, such as Java and C++, were not expected to be commonly used.
2.3. GOALS AND HYPOTHESES

Hypothesis 2.2. History of use, legacy code, and trust are the reasons why programming languages are chosen.

2. What tools are used to support development activities and why?

Hypothesis 2.3. Software engineering tools are not commonly used in scientific software development.

Hypothesis 2.4. Tools are not used due to lack of trust and unclear benefits of adoption.

3. Does development tend to be a team or individual activity?

Hypothesis 2.5. Scientific software is more commonly produced by individuals or small groups.

4. How large are the intended user bases and are the users programmers?

Hypothesis 2.6. Smaller user bases of less than 10 people are more common for scientific software.

Hypothesis 2.7. Users of scientific software are more likely to not have programming experience.

5. What documentation is produced and why?

Hypothesis 2.8. Documentation is not commonly produced in scientific software development.

Hypothesis 2.9. Documentation is not produced in scientific software development as the time and effort required is considered prohibitive when compared with benefits which are indirect and not immediately obvious.

6. What testing/verification is performed and why?

Hypothesis 2.10. Testing and verification is not commonly performed in scientific software development.

Hypothesis 2.11. Testing and verification is not performed in scientific software development due to the time and effort required, as well as a lack of available test data to conclusively demonstrate correctness.
7. What are considered to be the most important non-functional requirements from a developer’s point of view?

**Hypothesis 2.12.** The most important non-functional requirements for scientific software developers are performance, maintenance, portability, reliability, functionality, and traceability.

## 2.4 Methodology

The methodology used to conduct this survey is described here, along with details regarding the participants and survey contents.

The ethics approval form for this study is included in Appendix A.

### 2.4.1 Survey conduct

The survey was conducted on line through the Australian National University Polling Online (APOLLO) system*. The survey was run from August to September in 2009.

Subsequent to the survey being conducted, further literature has been published regarding the practices used for scientific software development. The results of this survey are supported by the subsequent research, and Section 2.7 summarises the subsequent literature and compares the results of this survey to those of this subsequent research.

Disclosure of all personal information was optional, and respondents’ answers were only saved once the entire survey was completed.

Prior to the full survey, a limited initial survey was run. This was done to test and streamline the survey, to make it as easy to complete for respondents as possible. As a result of this pilot study, one change was made to the content of the survey. In the questions relating to the importance of non-functional requirements (Section 2.5.8), performance and traceability were added to the list of non-functional requirements to be rated.

### 2.4.2 Participants

Participants were elicited from various scientific research communities. Scientists from the Australian National University in Canberra, Australia, and École Normale Supérieure in Paris, France, were contacted directly via email, and many

---

*http://apollo.anu.edu.au, accessed 1st February 2014*
indicated they forwarded the survey on through their own contacts and mailing lists. The survey was also distributed on a number of scientific computing email mailing lists.

Participants were required to have been directly involved in the development of scientific software applications. Sixty responses were received.

2.4.3 Survey contents

Seven areas were covered in this survey: demographics, programming languages, tools, development teams and user bases, documentation, testing and verification, and finally non-functional requirements. The full survey used is included in Appendix B.

Basic respondent demographics were collected to determine the spread of respondents to the survey.

The popularity of different programming languages, and the reasons that drive the selection of programming languages, was investigated as this selection affects which software engineering techniques would be of use to scientific software developers.

Current use of tools by scientific software developers to support their activities was covered. Past papers have presented differing views on the use of tools, ranging from a lack of tool use in the scientific software development community [127] to specific case studies where many tools have been used [2, 85]. Determining the popularity of tools and the reasons why they are used will help us determine if and where tool usage should be increased, and what kinds of tools may be applicable to scientific software development.

An overall impression of development team sizes, user base sizes, and user base programming experience was collected. These questions were included to see if there are any trends for these characteristics in scientific software, which again affects the software engineering techniques which may be applicable to scientific software development.

The negative opinion of documentation in scientific software has been remarked upon previously by Segal [109, 111], and as such the production of documentation is investigated in this survey. Identifying reasons why scientific software developers do or do not produce documentation will help us identify more appropriate documentation types and methods for scientific computing.

Testing and verification have been identified as areas where scientific software developers and users perceived importance is greater than their perceived
understanding [51]. In this survey, testing and verification activities performed by scientific software developers was also investigated, along with reasons for and against performing such activities. This information can be used to identify areas where testing and verification of scientific software development could be improved. Also, strategies to encourage uptake of testing and verification activities amongst scientific software developers can be identified.

Finally, the importance of various non-functional requirements such as functionality, performance, and usability was investigated. This information would again help identify the most appropriate software engineering methodologies for scientific software. It may also identify areas which scientific software developers currently do not, but perhaps should, consider important. The benefits of improving these areas may benefit scientific activities as a whole.

2.5 Results and discussion

The results of each of the topics covered in this survey will be reported and discussed in turn. These topics, described in greater detail in 2.4.3, are:

1. Demographics
2. Programming languages
3. Development tools
4. Development teams and user base characteristics
5. Documentation
6. Testing and verification
7. Non-functional requirements

This section also discusses threats to the validity of this survey, along with a summary comparing the results of this study to previous research.

2.5.1 Demographics

A total of 60 survey responses were received. The survey respondents were spread worldwide. Of the 47 survey respondents who chose to disclose their country, there were roughly even contributions from North America (17), Australia and Asia (14), and Europe (16). A more detailed breakdown is shown in Figure 2.1.
2.5. RESULTS AND DISCUSSION

Respondents were also asked optionally the organisation that they worked for, and 46 respondents provided this information. A summary of the types of organisation which they work for is shown in Figure 2.2. The types of organisation are universities including all educational and teaching institutions, industry and other companies, research organisations, and government departments and services.

Figures 2.3 and 2.4 show the highest level of education attained and the area in which that education was completed for the survey respondents. All 60 respondents provided information for the highest level of education, however only 56 respondents provided usable information regarding their field of study. Some respondents listed more than one area of educational study.

Of interest is that only 13, or 23%, completed studies in software engineering or computing-related fields. The majority of those who completed education in software engineering or computing had masters degrees. In comparison, most of those in scientific areas have doctorates.

Seven of the respondents who completed studies in software engineering or computing provided their position within the organisation that they worked for. Of those seven, three stated their roles as being a software programmer, devel-
CHAPTER 2. SCIENTISTS DEVELOPING SOFTWARE

Figure 2.2: Type of organisation worked for

Figure 2.3: Highest level of education completed
2.5. RESULTS AND DISCUSSION

In comparison, of the 38 respondents who did not complete studies in software engineering or computing who also provided their position, only four stated similar roles. This larger proportion of respondents in software programming, development, or engineering roles may explain the comparative lack of doctorates amongst the respondents who completed studies in software engineering or computing, as these people focus more on software development rather than scientific research.

The amount of programming experience of the survey respondents ranges from two to forty years of programming experience. There is a wide spread in the number of years of programming experience, with a median of 11. Figure 2.5 shows a breakdown of years of programming experience.

2.5.2 Programming languages

As shown in Figure 2.6, the most commonly used programming languages are C and C++, with 31 and 29 out of 60 respondents respectively. Perl (23), Java (20), Python (16), Fortran (10), and R (7) are also well represented. Other programming languages were mentioned by a couple of respondents, including C#, PHP and Matlab. The number of respondents indicating they used Fortran
was lower than expected, although the reasons for this discrepancy compared to other studies [37, 99] are not clear. Overall, these results do not support Hypothesis 2.1, with no type of programming language dominant.

Figure 2.7 shows the reasons why respondents choose to use certain programming languages. The four most common reasons provided by respondents were cross-platform compatibility, developer experience, features, and ease of use. Developer experience includes elements of the hypothesised reasons of history of use and trust, however neither of these were explicitly mentioned by respondents. Legacy code was the fifth most common reason. Overall, although Hypothesis 2.2 is not entirely supported by these results as the expected reasons of history of use, legacy code, and trust were not the most common reasons why programming languages are chosen, these reasons do appear within the five most common reasons for choosing programming languages.

In other studies, legacy code, trust, experience, and familiarity [72, 105] have been identified as influences on the choice of programming languages for scientific software. In this survey, similar reasons of developer experience, ease of use, and legacy code were each chosen by approximately 50% of respondents.

Overall, the qualities of the language (cross-platform compatibility, features,
2.5. **RESULTS AND DISCUSSION**

![Bar chart showing programming languages used](image1)

**Figure 2.6: Programming languages used**

![Bar chart showing reasons for choice](image2)

**Figure 2.7: Reasons for Choice of Programming Language**
ease of use, performance) were more common reasons for the choice of programming language than outside factors (developer experience, favourite language, required, only language known).

2.5.3 Development tools

When asked whether any tools or programs were used to aid in development, 50 out of 60 survey respondents indicated that there were tools or programs used. Figure 2.8 shows a breakdown of what kinds of tools were mentioned by the survey respondents. These tool types were determined from the specific tools which survey respondents listed when asked to list tools which they used.

These results do not support Hypothesis 2.3, that software engineering tools are not commonly used in scientific software development, as the majority of respondents indicated tools or programs were used to support development. However, when the types of tools which are actually used are investigated, it can be seen that many tool types are not commonly used. Only version control and Integrated Development Environments (IDEs) are commonly used by participants.

The lack of use of tools by scientists has been identified previously by Wilson [127], and is still an issue in more recent publications[126]. The results of this
survey appear to indicate that the situation is not as dire as implied by other literature both prior and subsequent to this survey. However, with only 29 out of 60 respondents indicating they use version control tools, it is clear that further progress could still be made. Use of tools to assist in bug/change tracking and testing is also very low.

The high number of respondents indicating they commonly use an IDE appears to be at odds with observations made by Carver et al. [16], where there was a lack of IDE usage and 'experienced developers tended to dislike the rigidity they felt most IDEs imposed on their development activities'. However, four of the five projects which they studied targeted parallel super computer hardware platforms, which leads to different requirements amongst developers.

The lack of use of bug/change tracking tools and build tools may be explained by Morris [90], as these types of tools are not needed for prototype or throwaway code which is commonplace in scientific software. Scientists often start programming with these types of code [90], and low usage of the aforementioned tools carries over when scientists move on to develop larger and more complex codes.

The survey also asked why the tools were used. A summary of the reasons given by the 46 respondents who answered this question and had indicated that tools or programs were used is shown in Figure 2.9. It should be noted that the reasons provided are a mixture of reasons for using a tool at all, and reasons for using a particular tool over another. Of special interest are the 12 respondents who described the use of version control as 'required' or 'mandatory' for large-scale, multi-developer, and/or distributed development projects. It is unknown whether these comments mean that the use of version control is enforced (and that these respondents would not otherwise use version control), or whether version control is considered essential for such projects (and that it would be impossible to run such large-scale, multi-developer, and/or distributed development projects without version control). As tools were commonly used by participants, these reasons for using tools do not provide much insight in regard to Hypothesis 2.4.

2.5.4 Development teams and user base characteristics

As can be seen in Figure 2.10, most of the survey respondents develop software either alone or in a small team of developers. Few of the respondents often or always develop software in teams comprising seven or more members. This supports Hypothesis 2.5. Due to limitations in the survey software used, the rating for each category of development team size was independent of the others.
For example a single respondent could answer ‘always’ for all four categories of development team size, although conceptually this does not make sense. The responses were manually checked for this kind of anomalous data, and one such result for development team size was discarded.

From Figure 2.11, there is a slight tendency for the intended user base size to be towards individual and small group user base sizes compared to larger user bases, which supports Hypothesis 2.6. Figure 2.12 shows there is a very slight tendency towards user bases being comprised entirely or partially of users with programming experience, rather than solely of users without programming experience as originally expected (Hypothesis 2.7). Due to the limitations of the survey software described previously, one result for user base programming experience was discarded.

2.5.5 Documentation

Figure 2.13 shows the number of respondents who indicated they produce certain types of documentation. The most common type of documentation produced by respondents was comments in the code, selected by 51 out of 60 respondents. At the other end of the scale, requirements documentation is the least commonly
2.5. RESULTS AND DISCUSSION

Figure 2.10: Development team sizes

Figure 2.11: Intended user base sizes
produced type of documentation with only 18 respondents indicating that they commonly produce such artefacts. In general, the results do not support the hypothesis that documentation is not commonly produced in scientific software development (Hypothesis 2.8).

The comparative lack of documentation for requirements and design matches the experiences described by Heaton et al. [54], Pawlik et al. [98], and Sanders and Kelly [104]. In the study performed by Sanders and Kelly [104] ‘none of [the] interviewees created an up-front formal requirements specification’, and ‘Only one interviewee ... demonstrated his design documentation’. This lack of explicit requirements and design documentation matches those of agile approaches [13]. In agile approaches, working software is emphasised over documentation [87], although the documents that are produced should be both necessary and sufficient [96]. In non-scientific software application development, agile approaches have become more widely used, but have not entirely replaced more traditional software development methodologies [123].

In this survey, 42 out of 60 respondents indicated they commonly produced user manuals and/or guides. However, as seen in Section 2.5.4, only 25 respondents indicated they often or always produced software for larger groups of up to
100 people and/or an international community of users. Therefore, at least 17 respondents are commonly producing user manuals and/or guides for themselves or users in their local area. This is at odds with Sanders and Kelly [104], who noted that none of their interviewees ‘produced user documentation unless their software was intended for use outside their research group’.

Only three respondents in this survey mentioned documentation being required by or compulsory for users or customers. All of the types of documentation that were defined in the survey were commonly produced by at least 18 of the 60 respondents. Therefore, scientific software developers produce documentation even though it is not required or compulsory. This contrasts to Segal [111] who identified that professional end user developers (non-software domain specialists who produce software for specific purposes) ‘did not voluntarily produce documentation, apart from the occasional user guide’.

The survey respondents were also asked why documentation is or is not commonly produced for scientific software applications which they have worked on. From the 50 responses to this open-ended question, a range of reasons both for and against the production of documentation were identified using the qualitative data analysis technique of coding [108].
Figure 2.14: Reasons for and against production of documentation

The most common reasons for and against the production of documentation and their popularity are shown in Figure 2.14. The reasons provided for why documentation is not produced supports Hypothesis 2.9, with the time and effort required being the most common reason provided. Note that these reasons are not mutually exclusive and are not complete. Many respondents provided both reasons for and against the production of documentation. Some reasons were only provided by a single respondent and are not included above.

2.5.6 Testing and verification

Figure 2.15 shows the number of respondents who stated they commonly perform the listed types of testing and verification. Contrary to expectations, all respondents indicated they performed at least one testing or verification technique, with most testing and verification activities being performed by at least a third of respondents, disproving Hypothesis 2.10.

As can be expected from the low numbers of respondents commonly producing requirements and design documentation, as shown previously in Figure 2.13, verification against specified requirements and design are the least common types of testing.
Interestingly, 38 out of 60 respondents stated they commonly performed unit tests, but only 23 stated they commonly performed integration testing. This result agrees with Heaton et al. [54] which reported that unit testing was more commonly performed than integration and regression testing.

The results also indicate that software requirements may be captured using methods other than a requirements specification document as 12 of the respondents, who stated they commonly performed verification against requirements, did not state that they commonly produced requirements specifications (Section 2.5.5). This indicates that requirements are either not captured, or are captured in a format other than a specific and distinct requirement specification. Test cases and comparisons with other models and empirical data may be viewed as the software requirements, and matching known or prior data may be considered as requirements.

This survey agrees with Kelly and Sanders [71], who identified that ‘testing remains the most commonly used quality assessment activity for software of any kind’. In these results, the four most common activities involve comparisons with known, trusted, or prior data. Verification and peer review were three of the four least commonly performed activities.
Also described by Kelly and Sanders [71, 104] is the problem of determining the success of a test in scientific software. This problem stems from the difficulty of finding appropriate oracle data, or data to which test results can be compared, and then being able to identify the cause of problems if differences to the test data are found. However, in this survey two of the three most common testing and verification activities were comparison with empirical data and comparison with other models. This indicates that, regardless of the problems with this type of testing activity, it is still widely used by scientific software developers.

Segal [111] identified that a characteristic of professional end user development practice 'is the lack of any disciplined testing procedure'. These results support this claim, as although a large number of respondents indicated they commonly perform testing and verification activities, there was a low number of respondents using bug/change tracking or testing tools (Section 2.5.3).

As was the case for documentation, respondents were asked why testing is or is not performed for scientific software applications which they have worked on. This question was answered by 49 respondents, and the main reasons were extracted from the answers using the qualitative data analysis technique of coding [108]. Figure 2.16 shows these results.

Figure 2.16: Reasons for performing testing and verification
Only a few respondents gave reasons as to why testing is not performed, those reasons being lack of management support, applications being considered to be not large or complex enough to warrant certain types of testing, or asserting that it is usually fairly clear whether the software is working as intended. These reasons do not match with those hypothesised (Hypothesis 2.11). However, all of the respondents who gave reasons as to why testing is not performed still indicated that they commonly performed at least two types of testing and verification.

Again, the reasons provided above are not mutually exclusive and are not complete. The reasons provided are a mixture of reasons for performing testing and verification, and reasons for performing a particular testing or verification activity over another.

### 2.5.7 User and stakeholder confidence

The survey also included the more general question 'Are there any other ways in which the confidence of users and stakeholders is increased in the scientific software applications, so that they can feel they can safely rely on the software? If so, please briefly describe these methods'. Twenty-nine respondents provided an answer to this question.

Types of techniques used were identified from the open-ended answers to this question using coding [108]. Multiple techniques could be included in a single answer. The results are shown in Figure 2.17.

Many of the methods mentioned by respondents focus on either showing the software working (examples of use on sample data, hands-on training, or use within the development group) or letting the users/stakeholders know how the software works (open source, scientific paper publication).

### 2.5.8 Non-functional requirements

The respondents were asked to rate a series of non-functional requirements on the following 5-point scale:

1. very unimportant
2. unimportant
3. neither
4. important
Figure 2.17: Methods used to increase user and stakeholder confidence

5. very important

This scale was chosen so that the relative importance of non-functional requirements could be determined from respondents’ answers. A straight ranking of non-functional requirements would only indicate how important respondents considered each non-functional requirement in comparison to others, but would not provide any information regarding how important a non-functional requirement was overall. The neutral response of ‘neither’ was included as some respondents may not consider a non-functional requirement or are unaware of it.

Non-functional requirements from the Software Requirements Specification Data Item described in MIL-STD-498 [119] were used and are as follows:

1. Availability (the ability to be accessed and operated when needed)
2. Flexibility (the ability to be easily adapted to changing requirements)
3. Functionality (the ability to perform all required functions)
4. Maintainability (the ability to be easily corrected)
5. Portability (the ability to be easily modified for a new software/computing environment)
6. Reliability (the ability to perform with correct, consistent results)

7. Reusability (the ability to be used in multiple applications)

8. Testability (the ability to be easily and thoroughly tested)

9. Usability (the ability to be easily learned and used)

The MIL-STD-498 was chosen as it provides a clearly specified definition of non-functional requirements. To this list, two more non-functional requirements were added:

10. Performance (the ability to run using a minimum of time and/or memory)

11. Traceability (the ability to link the knowledge used to create the application through to the code and the output)

These two additional non-functional requirements were added based on the responses from the initial pilot survey identified in section 2.4. The descriptions of each non-functional requirement were provided in the survey.

Figure 2.18 shows the rated importance of the non-functional requirements as a percentage of total responses, sorted by number of very important ratings. Table 2.1 lists the non-functional requirements in descending order of combined important and very important ratings.

Other studies have used various definitions for non-functional requirements which differ slightly from the MIL-STD-498 [119]:

- Correctness [16, 71] combines functionality and reliability from the MIL-STD-498.

- Portability has been defined more specifically than in the MIL-STD-498, as ‘The ability to easily port a project to new machines’ [16] and ‘Codes must run efficiently on multiple machines’ [7].

Reliability was considered to be the most important non-functional requirement overall, with 83% of respondents rating it as very important, and the remainder all rating it as important. Functionality also rated very highly, with 65% rating it as very important and 30% rating it as important. These two results corroborate previous results from Kelly and Sanders [71], in which ‘the singular importance of correctness’ for scientific software was identified, and Carver et al. [16], where the most highly ranked project goal was correctness.
Figure 2.18: Importance of non-functional requirements as rated by respondents

Table 2.1: Combined *important* and *very important* ratings for non-functional requirements

<table>
<thead>
<tr>
<th>Ranking</th>
<th>Requirement</th>
<th>Combined <em>important</em> and <em>very important</em> ratings (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Reliability</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>Functionality</td>
<td>95</td>
</tr>
<tr>
<td>3</td>
<td>Maintainability</td>
<td>90</td>
</tr>
<tr>
<td>4</td>
<td>Availability</td>
<td>87</td>
</tr>
<tr>
<td>5</td>
<td>Performance</td>
<td>79</td>
</tr>
<tr>
<td>6</td>
<td>Flexibility</td>
<td>77</td>
</tr>
<tr>
<td>7</td>
<td>Testability</td>
<td>75</td>
</tr>
<tr>
<td>8</td>
<td>Usability</td>
<td>63</td>
</tr>
<tr>
<td>9</td>
<td>Reusability</td>
<td>62</td>
</tr>
<tr>
<td>10</td>
<td>Traceability</td>
<td>54</td>
</tr>
<tr>
<td>11</td>
<td>Portability</td>
<td>52</td>
</tr>
</tbody>
</table>
2.5. RESULTS AND DISCUSSION

Portability received the highest number of unimportant ratings for any non-functional requirement (11), and the lowest combined proportion of important and very important ratings (52%). This result differs from a previous study by Carver et al. [16] where portability ranked third below correctness and performance but was ‘the only one that was ranked as having ‘high’ importance by all of the teams studied’. This difference may be explained by the fact that four of the five projects studied by Carver et al. targeted parallel super computer hardware platforms, whereas this study takes a much broader spread of scientific software applications.

Traceability received the second lowest combined proportion of important and very important ratings (54%). Previous studies have identified the importance to users of scientific software applications of transparency and traceability, particularly in regard to commercial and open source scientific software [105, 104]. In comparison, this survey shows it is less of a consideration for developers of scientific software.

Reusability received the lowest proportion of very important ratings (20%), and the third lowest combined proportion of important and very important ratings (62%). This helps explain the lack of documentation which was noted in Section 2.5.5. Developers do not anticipate coming back to the system to reuse part or all of it in the future, so the perceived need to document and record decisions made during the development process is low. According to Segal [109, 111] code reuse is rare and code is not generally designed for reuse, which agrees with the low importance that scientific software developers place on reusability in this survey.

An interesting result are the ratings for usability. When ranked in order of proportion of very important ratings, it is the third highest with 43%. However, when ranked in order of proportion of important and very important ratings combined, it is the 4th lowest with 63%. Therefore, although a comparatively large proportion of respondents consider it to be very important, there is also a large proportion of respondents who do not consider it important (in other words, consider it very unimportant, unimportant, or neutral). This indicates that the importance of usability is more split than other non-functional requirements - it is either very important or not important to scientific software developers, with a smaller proportion of developers considering it moderately important compared to other non-functional requirements.

Figure 2.19 shows how often respondents developed for particular user base sizes, with respondents broken down according to how important they rated usability. As an example, the left-most set of four columns shows that 26 respon-
dents rated usability as *very important*. The columns in this set of four each correspond to a different intended user base size - from left to right, a single user, less than 10 users, less than 100 users, or more than 100 users. Finally, each column is broken down into how often the respondents developed for that particular user base size - never, rarely, sometimes, often, or always. In the case of the 26 respondents who rated usability as *very important* (the left-most set of four columns), two respondents always, six often, seven sometimes, seven rarely, and four never developed for an intended user base of single user (the left-most column in the left-most set of four columns). Figure 2.20 similarly shows how often respondents developed for particular user base programming experiences, with the intended user base programming experience replacing the intended user base size.

Figure 2.19 shows that for important ratings of usability there is a tendency towards larger user base sizes, and for neutral and unimportant ratings of usability there is a tendency towards smaller user base sizes. However, for *very important* ratings of usability there is no tendency towards either large or small user base sizes. Figure 2.20 shows that respondents who rated usability as *important* or *very important* are more likely to develop for users who do not have programming experience, whereas those respondents who rated usability as *neutral* or *unimportant* are more likely to develop for users who have programming experience. These results agree with Sanders and Kelly [104], where usability was identified as being given priority when the developers 'were concerned about their user interface and understood that their users' knowledge didn't overlap their own'.

The result for maintainability is opposite that of usability - when ranked in order of proportion of *very important* ratings it is the third lowest (25%), but for *important* and *very important* ratings combined it is the third highest (90%). This indicates that, although maintainability is viewed to be *important* or *very important* by nine out of ten respondents, this importance is secondary to other non-functional requirements. This may be as maintainability affects future use and development of the software, but does not immediately impact the use of the software for scientific purposes. This result matches that of Carver et al. [16], where maintainability was the fourth highest ranked project goal, behind correctness, performance, and portability.

The remaining four non-functional requirements, availability, flexibility, performance, and testability all had *important* as their most common response and ranked in the middle when ordered both for *very important* responses alone or
2.5. RESULTS AND DISCUSSION

Performance has been identified as important in scientific software [37, 63], although other studies have described it as not the singular be-all and end-all requirement [72]. As mentioned previously, performance is balanced with portability, maintainability, and ease of programming [7]. The results of this survey broadly agree with Carver et al. [16] who identified performance as the second most highly ranked project goal behind correctness and ahead of portability and maintainability, however this disagrees with the importance placed on performance by Faulk et al.[37] and Jones and Scaffidi[63].

Overall, Hypothesis 2.12 is only partially supported by these results. As expected, reliability and functionality were the most important non-functional requirements for scientific software developers. However, although they were considered to be important, performance and maintainability were secondary to other non-functional requirements such as availability. Finally, contrary to the hypothesis, portability and traceability were not considered as important as other non-functional requirements.

Figure 2.19: How often respondents developed for particular user base sizes, respondents categorised by usability rating

important and very important responses combined.
CHAPTER 2. SCIENTISTS DEVELOPING SOFTWARE

Figure 2.20: How often respondents developed for users with, without and both with and without programming experience, respondents categorised by usability rating

2.5.9 Threats to validity

The sample size of this survey, 60 respondents, compares favourably against other recent studies [16, 90, 104].

There is a larger number of respondents who completed their education in Bioinformatics and Biology-related fields compared to other fields, which skews the results towards the practices and views of developers in those fields. This skew affects the external validity of this study, which could explain the differences between the results of this study and other previous studies, which used samples of software developers from other scientific fields. The lack of information regarding the level of software engineering and computing education of scientific software developers also means that it is unknown whether the proportion of respondents with formal education in software engineering or computing is representative or not.

As mentioned in Section 2.5.3, the examples listed in the question ‘Were any tools or programs such as IDEs, automated test environments, or version control software used to aid in the development of the application?’ may have influenced
respondents to more readily list those types of tools in their answer, leading to the much greater number of respondents listing IDEs, version control, and to a lesser extent testing software tools.

The reasons for using tools (Section 2.5.3) are not directly associated with particular tools or types of tools. A similar problem occurs for reasons for and against producing documentation (Section 2.5.5) as well as for reasons for performing testing and verification activities (Section 2.5.6). Although this could be avoided by asking for reasons for each type of tool, documentation, or testing and verification activity, this would increase the length of the survey considerably. These areas could be addressed in greater detail in separate surveys focussed on each activity, or by conducting interviews. This would avoid having a single, overly lengthy survey.

As described in Section 2.5.4, the ratings for each of the categories provided for development team sizes, user base sizes, and user base programming experience, were independent - a single respondent could possibly answer always for all four of the development team size categories. This inability for an answer to one question to constrain the allowable answers to another question are inherent in the survey software used (see Section 2.4). Results were checked manually for any occurrences of this nature, with one result discarded for development team sizes and one result discarded for user base programming experience.

2.5.10 Comparison with previous research

A summary of the survey results as compared to notable past papers is shown in Table 2.2. Details of these comparisons have been explored in Sections 2.5.1 to 2.5.8. Those areas where the results of this survey disagree with the results of previous work are as follows:

1. Carver et al. [16] identified a lack of IDE usage, whereas in this survey 34 out of 60 respondents indicated they commonly used an IDE. Carver et al. also identified that portability ranked third below correctness and performance but was ‘the only one that was ranked as having ‘high’ importance by all of the teams studied’, whereas in this survey portability received the lowest combined proportion of important and very important ratings (52%). These differences may be due to four of the five projects which Carver et al. studied targeting parallel super computer hardware platforms, which have a different set of requirements and priorities than other scientific software applications targeting smaller hardware platforms.
Table 2.2: Survey results compared to past papers

<table>
<thead>
<tr>
<th>Survey Area</th>
<th>Carver et al. [16]</th>
<th>Kelly et al. [71]</th>
<th>Sanders et al. [104]</th>
<th>Segal [111]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool Use</td>
<td>✗</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Documentation</td>
<td></td>
<td>✓</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>Testing and Verification</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functionality</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maintainability</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usability</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reusability</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Traceability</td>
<td></td>
<td></td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>Portability</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

✓ Survey agrees with past paper
✗ Survey disagrees with past paper
(Blank) Past paper does not address this area

Note that availability and testability are not included as these have not been mentioned in previous papers. Flexibility is also not included as this has only been mentioned in conjunction with maintainability in previous papers.

2. Sanders et al. [104] identified the high importance of traceability, compared to the relatively low importance of traceability in these results. However, this result is for scientific software development, whereas the result of Sanders et al. is for scientific software use. This indicates a difference between the attitudes of those who develop and those who use scientific software - users want to see and understand how the results are generated, whereas developers already know the process as they created the software.

3. Segal [111] identified that scientific software developers ‘did not voluntarily produce documentation, apart from the occasional user guide’. In this survey most types of documentation were produced by 20 to 40 of the 60 respondents, and only three respondents indicated that documentation was required. Documentation is now produced for reasons other than it being ‘required’, such as for future maintenance purposes.

The results of this survey have not been compared to those of the survey Hannay et al. [51] conducted. Unlike this survey, Hannay et al. does not cover
specifics such as what types of tools, documentation, and testing and verification activities are used, instead covering how scientists learn about and the time spent working on developing and using software, and the perceived importance and understanding of software engineering concepts.

2.5.11 Summary of results compared to hypotheses

The results of this study compared to the hypotheses, which were originally presented in Section 2.3, are presented here. Each hypothesis will be restated and the relevant results briefly summarised.

**Hypothesis 2.1.** The most common programming languages in scientific software development are lower-level programming languages, such as FORTRAN and C. Higher-level languages, such as Java and C++, were not expected to be commonly used.

Hypothesis 2.1 was not supported by the results of this study, as no type of programming language was dominant over the others.

**Hypothesis 2.2.** History of use, legacy code, and trust are the reasons why programming languages are chosen.

Hypothesis 2.2 was partially supported, however the additional reasons of cross-platform compatibility features, and ease of use were not expected.

**Hypothesis 2.3.** Software engineering tools are not commonly used in scientific software development.

Hypothesis 2.3 was not supported, with the majority of respondents indicating that tools or programs were used to support software development. However, only IDEs and version control were commonly used, with other types of tools not commonly used by respondents.

**Hypothesis 2.4.** Tools are not used due to lack of trust and unclear benefits of adoption.

The results of this study do not provide much insight regarding Hypothesis 2.4, as the majority of participants indicated they used tools and provided reasons for the use of tools, rather than lack thereof.

**Hypothesis 2.5.** Scientific software is more commonly produced by individuals or small groups.
Hypothesis 2.5 is supported by the results of this study, with most respondents developing software either alone or in a small team of developers.

**Hypothesis 2.6.** Smaller user bases of less than 10 people are more common for scientific software.

Hypothesis 2.6 is partially supported by the results of this study. There is a slight tendency towards individual and small group user bases, rather than larger use bases.

**Hypothesis 2.7.** Users of scientific software are more likely to not have programming experience.

Hypothesis 2.7 was not supported, as there is a slight tendency towards user bases being comprised either of only users with programming experience, or both users with and without programming experience.

**Hypothesis 2.8.** Documentation is not commonly produced in scientific software development.

Hypothesis 2.8 was not supported, as most respondents commonly produced at least one type of documentation.

**Hypothesis 2.9.** Documentation is not produced in scientific software development as the time and effort required is considered prohibitive when compared with benefits which are indirect and not immediately obvious.

Hypothesis 2.9 was supported, with the time and effort required being the most common reason provided for the documentation not being produced.

**Hypothesis 2.10.** Testing and verification is not commonly performed in scientific software development.

Hypothesis 2.10 was not supported by the results of this study, as every respondent indicated they performed at least one testing or verification technique, and nearly all techniques being performed by at least one third of respondents.

**Hypothesis 2.11.** Testing and verification is not performed in scientific software development due to the time and effort required, as well as a lack of available test data to conclusively demonstrate correctness.
2.6. IMPLICATIONS

Hypothesis 2.11 was not supported, as the few respondents who provided reasons why testing is not performed gave the reasons of lack of management support, not being warranted due to the small size or simplicity of the program, or asserting that it is usually clear if the software is working as intended.

**Hypothesis 2.12.** *The most important non-functional requirements for scientific software developers are performance, maintenance, portability, reliability, functionality, and traceability.*

Hypothesis 2.12 is partially supported by the results of this study, with reliability and functionality being the two most important non-functional requirements. However, performance and maintainability were of secondary importance compared to other non-functional requirements, and portability and traceability were considered the least important overall.

2.6 Implications

This survey shows that the situation in scientific software development may have improved in recent years, and at least in certain areas is not as dire as presented in the existing literature.

- Although tool use is by no means widespread, version control and IDEs are used by around 50% of developers. This indicates that at least some tools have made some progress in being more widely used to support scientific software development.

- Likewise, particular types of documentation are produced by an average of 50% of developers, with some (user manuals and/or guides and comments in the code) more common and others (requirements specifications and design documentation) less so. Although the time and effort required for documentation is still seen as a hurdle, documentation is more widespread in scientific software development than suggested by previous literature.

- Testing and verification activities are also performed inconsistently, with most types of testing being performed by just under 50% of developers. Comparisons with empirical data or data from other models, unit testing, and regression testing are the exceptions to this.

Although there remains much room for improvement, the introduction of basic software engineering methodologies and techniques is no longer the primary or
most widespread issue in at least some areas of scientific software development. Given that scientific software development is, at least in part, now familiar with these aspects of software engineering, increased attention can now be paid to the introduction, development, and adaptation of other software engineering methodologies and techniques in order to improve scientific software development.

There are certain considerations to be made when exploring how software engineering can improve scientific software development:

- No one programming language dominates scientific software development, and newer programming languages such as C++ and Java coexist with older languages such as Fortran, with cross-platform compatibility, features, developer experience and ease of use being the most common reasons for selecting a programming language.

- The most commonly stated reason for using tools is to improve ease of coding.

- Reliability and functionality are the most important non-functional requirements for scientific software developers, with usability, maintainability, and availability also considered to be important.

- Scientific software is most often developed by a single person or a small team (2-6) of people, but can be intended to be used by any number of users with or without programming experience.

### 2.7 Subsequent literature

Although the area of scientific software development received much attention from 2007-2009, there has also been a number of studies in this field past that date. The following literature was published after the survey was performed.

Heaton [53] identifies four major problems which were self-reported by computational science and engineering developers: rework, performance issues, regression errors, and forgetting to fix bugs that were not tracked. A follow-up survey [54] identified that amongst research developers (developers who spent 30% or less of their time on software for external users), creation and use of requirements documentation was low, however other documentation was used frequently. A low to medium level of use was reported for design activities. Unit testing was highly used, however a wide variety of usage levels was reported for integration and regression testing. Finally, issue/bug tracking software was not used often.
2.7. SUBSEQUENT LITERATURE

Jones and Scaffidi[63] purport to investigate the obstacles and opportunities with 'visual and domain-specific languages in scientific programming', however their results include a significant amount of discussion regarding the problems faced by scientific software developers. Procedural constructs were often used rather than object-oriented constructs if possible. Four out of the nine participants described performance as a major concern. Maintenance problems were identified as being caused by a lack of version control in two projects, and a lack of documentation in five projects.

Pawlik et al.[98] investigates documentation practices in scientific software development, finding that the practices used differ from those formally recommended in software engineering. Requirements and specifications are usually not produced, and design and architectural documentation are created ad-hoc if they are created at all. Implementation details are mainly captured in code comments. Documentation for users, such as tutorials and examples of usage, is produced when requested. One particular reason provided for the lack of documentation was that for the scientists developing software, the activity of producing documentation was not valued and considered 'a mundane and boring task'.

Prabhu et al.[99] surveyed 114 scientists from various disciplines at Princeton University regarding their 'use of computation in research'. This survey found that Matlab, Fortran, C, C++, and python were the dominant programming languages. The 'debugging and testing methods employed were primitive', with more than half of the researchers not using any debugger.

2.7.1 Summary of subsequent literature

The subsequent literature shows some similarities as well as differences with the previous literature. Similarities occur in the following areas:

- Lower-level programming languages remains more popular than higher-level, more abstract languages [99], with procedural constructs preferred over object-oriented constructs [63].

- The use of tools varies, with issue and bug tracking software rarely used [53] and limited use of debuggers [99].

- The production and use of documentation varies. Requirements documents are consistently rarely created or used [54, 98], as is design and architectural documentation [98].
CHAPTER 2. SCIENTISTS DEVELOPING SOFTWARE

• Performance remains an important factor for some, but not all, scientific software developers [63]. However, there are a number of differences:

• Although version control is still lacking in some projects, it is used in a majority of projects [54].

• Some types of documentation are produced. Documentation for users is produced when requested [98], and documentation other than requirements are used frequently [54].

• Testing practices remain an issue [54, 99], however unit testing was found to be highly used [54]. In comparison, the use of integration and regression testing varies widely [53].

2.7.2 Comparison of results with subsequent research

The more recent research published after the survey was conducted collaborates many of the results. In particular, the following findings are supported by more recent literature:

• Some types of tools are more commonly used than previously expected, such as version control [54].

• Although requirements and design documentation remains rarely created, other types such as user documentation are relatively common [54, 98].

• Certain types of testing are commonly performed, such as unit tests. Although regression tests are commonly performed, integration tests are not, which agrees with the wide variation in use of these two types of testing found by Heaton et al. [53].

2.8 Conclusion

In summary, these results indicate that many of the issues of the development of software for scientific research purposes are no longer as significant as previously thought. The use of mainstream software engineering techniques in software development for scientific research purposes has improved greatly, and thus the challenges identified in the existing literature are on their way to being addressed.
2.8. CONCLUSION

Therefore, in the field of scientific software development, it is appropriate to move on from the introduction of basic software engineering techniques, and investigate the applicability of more advanced software engineering methodologies. Software for scientific research purposes should be as reliable and as functional as possible, while also being easy to use by scientists with varying levels of programming experience. The application of software engineering into the scientific software field must take these factors into account.

The results of this study and other literature indicate there still are some areas where software engineering could be of use in scientific software development. Documentation, testing, and verification could be made more easily accessible to scientific software developers. Tools and techniques which make it easier to perform coding tasks are more widely used. Scientific software developers are most interested in the reliability, functionality, usability, maintainability, and availability of their software programs.

Based on these factors, DSVLs were identified as an approach which would be of benefit to scientific software programming. DSVLs improve validation [8, 14] and documentation [56, 77], areas where further improvement of scientific software development is possible. DSVLs also improve reliability [55, 56, 73], maintainability [9, 120, 73, 74, 75, 86], and usability [9, 55, 74, 75], all of which are important for scientific software developers. Finally, DSVLs increase productivity [120, 56, 73, 77, 86] which would result in increased scientific output.
CHAPTER 2. SCIENTISTS DEVELOPING SOFTWARE
Chapter 3

Scientists and DSVLs

3.1 Introduction

As explored in previous literature, scientific software provides a unique challenge in that the people involved tend not to have formal software engineering training. The study presented in Chapter 2, along with other literature, shows that although some software engineering techniques are becoming more widely used in scientific software development, there remain some gaps. Documentation [31, 93, 98, 105, 109] and validation [68, 90, 93, 99, 104, 105, 110, 111] are two areas of ongoing concern. In addition, reliability, maintainability, and correctness were considered to be important by scientific software developers [16, 71, 93, 104, 111].

Based on these needs, DSVLs were identified as an approach which would be of benefit to scientific software development. DSVLs have been shown to improve all of the above areas - documentation [56, 77], validation [8, 14], reliability [55, 56, 73], maintainability [9, 120, 73, 74, 75, 86], and correctness [8, 14] - when applied to the development of software outside the area of scientific research. In addition, the productivity benefits of DSVLs [120, 55, 56, 73, 77, 86], and in particular the reduction in programming knowledge [86] and intellectual effort required between the conceptualisation and realisation (in a software system) of an idea [76], would also result in greater scientific output.

However, the benefits of DSVLs will remain unrealised if they are not accepted and used within the scientific software area. Thus far, research has focussed on producing new DSVLs and DSLs [12, 36, 91], with little analysis of the acceptability of the approach in the area of scientific software.

It should be noted that DSVLs would not be appropriate across all aspects of scientific software development. Although the benefits of DSVLs generally match
CHAPTER 3. SCIENTISTS AND DSVLS

the previously identified areas of issue and importance in scientific software development, in some areas of scientific software development the costs involved with implementing DSVLs could outweigh the benefits. In particular, the development of a DSVL generally involves a higher start-up cost than developing a system from scratch with a General Purpose Language (GPL) [61], but results in less effort spent on repeated similar tasks and maintenance [86]. Therefore, DSVLs would not be the most appropriate tool for developing smaller, one-off scientific software programs.

In order to address this lack of understanding, this chapter presents an empirical study which compares the acceptance of a DSVL tool with an existing non-visual software program for the purposes of environmental scientific computational modeling. Additional information was collected to identify reasons for the acceptance or rejection of the DSVL tool within the environmental scientific modeling community.

The scope of this study was limited to computational environmental scientific modeling for three reasons. First, the choice was made in order to narrow the focus of the study to a clearly defined subset of scientists, rather than attempting to cover the wide variety of different scientific fields and varying uses of software in the scientific community. Second, modeling in general is used in many scientific areas and forms a significant part of scientific research [43], and therefore generalisation of the results from this study to the wider field of scientific modeling would be valuable to multiple areas of scientific research. Third, analysis in computational environmental scientific modeling can involve the use of a single software model multiple times, each time with varied inputs of parameters, data, and other settings, and as previously discussed DSVLs would be of greater benefit in this type of software use compared to smaller, one-off scientific software programs.

First, existing literature covering the known benefits of DSLs and DSVLs and the current use of DSLs and DSVLs in environmental scientific computing are summarised. Second, the research questions and hypothesis of this study are detailed. Third, the methodology used in this study is described. Fourth, the results of the study are presented and discussed. Finally, the implications of the results from this study are explored.
3.2 Existing literature

A Domain-Specific Language (DSL) is a language designed to be relevant to a single problem domain or area [42, 71]. DSLs are used to describe and represent situations, problems, or solutions within their specific domain. In order to do this, a DSL uses ideas, concepts, and terms which are particular to the targeted domain. Two examples of DSLs are HTML (which describes web pages), SQL (which describes relational database queries).

This limited domain application is distinct from General Purpose Languages (GPLs), which can be programming languages, such as C and Java, or modeling languages, such as the Unified Modeling Language (UML) [102]. A GPL can be used to describe many different solutions and implementations through the use of generic language.

A Domain-Specific Visual Language (DSVL) is a DSL which uses a visual diagramming interface. A DSVL retains the same features of a DSL - namely, using ideas, concepts, and terms particular to the targeted domain. Instead of using a textual representation, a DSVL uses a graphical representation, usually consisting of a variety of symbols with corresponding text labels.

The existing literature surrounding DSLs and DSVLs is explored from three aspects - firstly, the benefits which DSLs and DSVLs bring to software development, secondly the acceptance of DSLs and DSVLs, and finally current use of DSLs and DSVLs in scientific fields.

3.2.1 Benefits of DSLs and DSVLs

DSLs and DSVLs have been created and used for many different purposes. Much of the published literature concentrates on the development, functionality, and use of said DSLs and DSVLs. Although much of the research refers to non-visual DSLs, it has been shown that the benefits of DSVLs mirror those of DSLs [70]. DSLs have been shown to improve productivity, reliability, documentation, usability, and correctness. Each of these benefits will be explored in turn.

Projects which have used DSLs have demonstrated improved productivity. Although DSLs involve an up-front cost in design and development of the DSL itself [61], once the DSL is built there are benefits in development [55, 56, 73, 86] and maintenance [120, 56, 73, 86] productivity which outweigh the initial cost. Two of the features of DSLs which leads to improved software development productivity are a reduction in the amount of programming knowledge required and
a reduction in the intellectual effort required to progress from conceptualisation to production of a software system. Once a DSL is constructed, non-software specialists are able to use the language of the domain to build the desired system using the DSL. This means that non-software specialists are not required to learn the intricacies of a software programming language [86] in order to build systems using DSLs. Due to being able to express themselves using the language of the domain, DSLs reduce the effort required to translate concepts within the domain to corresponding concepts within software programming [76].

Maintenance productivity is improved through the use of DSLs. DSLs improve correctness and reliability by reducing the number of software bugs introduced in the coding stage, as common or similar code components are managed through the DSL rather than by developers directly. The reliability improvements of DSLs have been demonstrated in several projects [55, 56, 73]. DSLs enable improved validation and optimisation techniques, including applying these at the domain and conceptual level rather than just at the programming code level [8, 14], and these techniques also improve correctness and reliability. Maintenance activities have also been shown to be easier and quicker with DSLs [9, 120, 73, 74, 75, 86].

DSLs have also been shown to improve the quality of documentation for software projects [56, 77]. Much of this benefit is due to DSLs being self-documenting to a large degree, as the specification of the program is primarily written in the language of the domain rather than in programming code. This reduces the need to produce separate documentation which explains the link between the programming code and the domain it is written for. As another effect of using the language of the domain rather than programming code, programs produced using DSLs are easier to understand [75, 74]. Generation of programming code means that the resultant program is more consistent, shorter, and clearer [9].

### 3.2.2 Acceptance of DSLs and DSVLs

Little research has been performed into the acceptance of DSLs and DSVLs, as evidenced by Gabriel et al.[44] in their systematic literature review. Only three out of 36 papers reported using domain experts in the assessment of a DSL. Over 80% of papers provided no evaluation of usability, and only one of the 36 papers reported a formal usability analysis.

Although the importance of usability evaluation has been argued and justified [6], this has not yet been performed. The use of interactive user-centred design and empirical evaluation and creation of metrics and methodologies for usability
3.3. GOALS AND HYPOTHESES

evaluation has been proposed [6]. Others have claimed to have used some sort of user-centred design with a continuous evaluation process at each iteration of the design and development process, but the details and results of these evaluations have not been presented [60]. Similarly, other DSVLs have reportedly been subject to usability trials, however these trials are not explained nor are the results presented [50]. Finally, a framework for characterising languages and a questionnaire to assess languages has been developed, however no evidence has been found to support whether it has been used for evaluation [52].

3.2.3 DSLs and DSVLs for scientific modeling

There are many instances of DSLs and a few DSVLs for use in scientific modeling. Research in this area concentrates on the applicability and ability of the DSL or DSVL to model aspects of the target domain. DSLs and DSVLs have been asserted to provide many improvements for scientific programming, but little analysis of the acceptability of DSLs or DSVLs by scientists has been performed.

Asserted benefits of DSLs and DVSLS within scientific fields include improved correctness, documentation, ease of use [12], simplicity, flexibility, capability, modularity, transparency, efficiency, re-usability, adaptability, and communicability [36]. Simile [91] and Stella [21] are both visual modeling tools and have both been used to implement ecological simulation models, however neither place particular constraints on what the elements and links within the language are allowed to represent. Simile is asserted to improve documentation and have an intuitive user interface. Stella is asserted to reduce the time required for development.

Finally, there has been some investigation of the obstacles and opportunities regarding visual languages and DSLs in scientific programming [63]. Although DSLs have been used for scripting and generating data visualisations, the use of a DSVL is not explored.

In summary, although the benefits of DSLs and DSVLs in scientific fields have been asserted, little evidence has been shown regarding either the benefits or acceptability of DSLs and DSVLs in scientific fields.

3.3 Goals and hypotheses

In order to improve the current understanding of the acceptability of the DSVL approach in the field of computational environmental scientific modeling, the
following research questions were posed:

1. Are DSVLs perceived differently from non-visual software interfaces by environmental scientific modelers, for the purpose of specifying an experimental scenario?

2. What factors affect why environmental scientific modelers do or do not accept the DSVL approach?

The hypotheses for these research questions were as follows:

**Hypothesis 3.1.** *The DSVL tool is more acceptable than a baseline tool.*

**Hypothesis 3.2.** *The diagrams drawn by environmental scientific modelers to explain their experimental scenarios are similar to those of software engineers.*

It was expected that these diagrams would be roughly equivalent to static model diagrams drawn by software engineers, such as UML class diagrams and UML dataflow diagrams[102]. The diagrams drawn by environmental scientists would contain elements and links in a manner similar to the software engineering static model diagrams, and therefore could be used as the basis for a DSVL in order to specify software simulations of their experimental scenarios.

**Hypothesis 3.3.** *Environmental scientific modelers more easily understand and communicate the specified experimental scenario using the DSVL tool, compared with a baseline tool.*

### 3.4 Methodology

The methodology used to conduct this study is explored in this section, including details regarding participants, the conduct of the experiment and the activities which the participants performed, a description of the tools which were compared, and the approach used to perform the statistical analysis of the collected data.

The ethics approval form for this study is included in Appendix C.

#### 3.4.1 Experimental conduct

The experiment consisted of one-on-one sessions, each of which lasted approximately one hour. In each session, participants were asked to perform a number
of tasks, which are described in Section 3.4.3. Sessions were conducted during June and July in 2011.

The study was advertised in an email mailing list with a two week response time limit. A reminder email was sent two days before responses closed.

3.4.2 Participants

Participants were elicited from postgraduate students and academic staff at the Australian National University (ANU). Participants were required to have a background or currently be involved in environmental sciences.

Nine environmental scientific modelers participated in this study. Five participants were postgraduate students, three were academics, and one was a research scientist.

Six participants had first-hand experience developing software for environmental models, with four of those participants having up to five years of modelling experience and the other two having between 16 and 20 years of experience. The remaining three participants used and participated in the conceptual and theoretical stages of producing environmental models, but were not directly involved in producing software code after their university-level education. DSVLs help enable non-software specialists to develop systems [86], and this study compares the acceptance of two differing approaches for scientists to specify a scientific software simulation. Therefore, these participants who currently relied upon or were involved with environmental models developed by others were also the intended target audience for the tools compared and thus also appropriate participants in this study.

Three participants indicated they had used visual or diagram-based tools for environmental modeling before, namely Stella [21], Netica*, Music†, and Vensim [32]. None had used either of the tools (described in Section 3.4.4) that were tested in this study.

The information sheet used for this study is included in Appendix D. The consent form, which all participants were required to complete prior to participation, is included in Appendix E.

---

3.4.3 Sessions and activities

Each session consisted of the following five activities:

1. Reading a one-page scenario summary
2. Drawing a diagram that described the participant’s view of the scenario
3. Guided walkthrough of the two ecological simulation tools
4. Survey of acceptability of the two tools
5. Semi-structured interview

Activity 1

The participant was asked to read a one-page scenario summary. Participants were allowed to ask questions to clarify their understanding of the scenario.

The scenario used was extracted from ‘Land-use change and subalpine tree dynamics: colonization of Larix decidua in French subalpine grasslands’ [3]. This scenario was originally simulated using the Functional Attributes in Terrestrial Ecosystems (FATE) vegetation stand dynamics model [89]. Pertinent information was condensed into a single one-page summary with the assistance of one of the scenario’s original authors. The summary provided to participants is included in Appendix F.

Activity 2

The second activity investigated research question 2 (Section 3.3). In this activity, participants were instructed to draw a diagram that described their view of the scenario, as if they were recording or explaining the scenario. Apart from this instruction, participants were given no guidance as to what their diagrams could or should contain.

Participants were provided with four pens of differing colour (black, blue, red, and green) and an A3 sheet of paper to draw on.

These diagrams were analysed regarding their form (how the information was presented) and function (what information was presented). This activity was performed before the participants were exposed to the two tools to avoid biasing their diagrams towards those elements represented in the tools.
3.4. METHODOLOGY

Activity 3

The third activity consisted of a guided walkthrough of the two ecological simulation tools, which are described in Section 3.4.4. To minimise the effect of bias due to ordering, the order in which the two tools were used was randomly allocated for each participant.

The participant was given control of the computer during this activity, and set up a software simulation of the previously described scenario using both of the tools. The guided walkthrough for each tool began with the participant being presented with the tool’s interface as seen on program start-up. The experimenter, who was experienced in setting up the scenario using both tools, provided verbal instructions to the participant, describing each step involved in setting up the scenario and how to perform those steps.

No time limit was placed on this activity, with participants proceeding until they had performed all of the required steps to set up the scenario using the current tool and had no questions with regard to using the tool. Generally, this activity lasted between 15 and 20 minutes total.

Activity 4

In the fourth activity the participant was asked to rate both tool interfaces. In order to measure acceptance of the tools to answer research question 1 (Section 3.3), the Technology Acceptance Model (TAM) [24] was used, with updated measurement scale items [25]. In the TAM, users rate their perceived usefulness and perceived ease of use of a subject technology. Such ratings have been shown to have a strong correlation with the user’s intention to use or acceptance of the subject technology [25]. The statements presented to the participants, or ‘measurement scale items’ as described by the TAM, were used without modification. Participants were asked to rate each of the statements considering environmental modeling as the ‘job’ referred to in the statements. The DSVL tool was described as the ‘diagram-based tool’, and the Baseline tool described as the ‘form-based tool’.

The measurement scale items for perceived usefulness are:

1. Using the [Tool] in my job would enable me to accomplish tasks more quickly
2. Using the [Tool] would improve my job performance
3. Using the [Tool] in my job would increase my productivity
4. Using the [Tool] would enhance my effectiveness on the job
5. Using the [Tool] would make it easier to do my job
6. I would find [Tool] useful in my job

The measurement scale items for perceived ease of use are:

1. Learning to operate the [Tool] would be easy for me
2. I would find it easy to get the [Tool] to do what I want it to do
3. My interaction with the [Tool] would be clear and understandable
4. I would find the [Tool] to be flexible to interact with
5. It would be easy for me to become skillful at using the [Tool]
6. I would find the [Tool] easy to use

As used by the TAM, each of the statements were rated on a seven point Likert scale [81]. For data analysis, the seven point scale was coded to the ratings 1-7, with Extremely Unlikely being 1 and Extremely Likely being 7.

Although there exist other more recent models which explain user acceptance, the original TAM was chosen as both of its scales measure aspects (perceived usefulness and perceived ease of use) of the system being analysed. The TAM does not include scales which measure external factors such as management and organisational support (as the Unified Theory of Acceptance and Use of Technology (UTAUT) [122] does), nor the opinion of people who influence the user's behaviour or are important to the user (as both UTAUT and the TAM2 [121] both do). As this study investigates the differences in acceptance due to differences in the tools rather than differences due to societal impacts, the original TAM was deemed to be the most appropriate measurement instrument.

In addition to perceived usefulness and perceived ease of use, perceived modeling utility was also measured to further investigate research question 2 (Section 3.3). The measurement scale items used were developed by the author. Modeling utility was defined as understanding, exploring, and communicating an experimental model. These measurement scale items are distinct from those used in the TAM, which measure perceived usefulness and perceived ease of use in general.

The six measurement scale items for perceived modeling utility are as follows, where [Tool] is replaced with the description of the appropriate tool:
3.4. METHODOLOGY

1. I would easily understand the scenario when using the [Tool] 

2. The [Tool] would give me a clear view of the scenario 

3. The [Tool] would allow me to explore the scenario easily 

4. I would be able to grasp the scenario structure easily using the [Tool] 

5. Using the [Tool] would make it easy to communicate the scenario with others 

6. I would easily understand a different scenario if it were presented to me using the [Tool] 

The questionnaire presented to participants is included in Appendix H.

Activity 5

The fifth activity, the semi-structured interview, was used to check, clarify, and expand upon the results of the other activities. Participants were asked to provide a verbal summary of their drawn diagram, queried about specific answers from the questionnaire in Activity 4, and asked for overall impressions of the tools.

3.4.4 Tools

The two tools compared in this study both provide an interface for users to specify an ecological simulation using the FATE model [89], in which the scenario provided was originally simulated. Although differing in the specific interface characteristics, both tools provide users with the ability to specify the required plant species groups (known as Plant Functional Types (PFTs)) and their characteristics such as lifespan, seed production, and seed distribution.

Baseline tool

The tool chosen for the baseline was LAMOS, developed by Lavorel et al. [78]. LAMOS is an existing tool used by environmental scientific modelers world-wide for various ecological simulations, including to simulate the effects of land-use change and tree dynamics [3], compare the sensitivity of landscape, fire, succession models [17], simulate feedback between fire and grass expansion [47], and simulate the effects of grazing regimes [22].
LAMOS is a modeling shell, and provides a form-based interface for specifying software simulations of ecological landscape models. This form-based interface operates in a similar manner to many commonly used programs, and involves users navigating through different drop-down menus to open different form windows, where the various parameters for the ecological software simulation can be specified.

Figure 3.1 shows a screen shot of the baseline tool. The current window open is the ‘Life history and seedpools’ menu for the PFTs present in the simulation.

The LAMOS interface can be divided into three areas: on the top are the menus and tool bars, on the bottom is the current experiment run information, and the remainder is the working window. In Fig. 3.1, the current working window is for the PFTs, with four tabs for different groups of the PFT parameters. There are additional windows for simulation parameters, map parameters, PFT dispersal, and disturbances. Each of these windows includes the appropriate tabs and parameters for the given category.

DSVL tool
The DSVL tool was developed and created specifically for this study, and provides an interface with functionality similar to others of its type, such as MetaEdit+ [69], metaCase [70], Marama [49], and Pouamau [129].

The GEMS framework developed by White et al. [124] was used to develop the DSVL. GEMS is built within the Eclipse IDE [33]. GEMS was chosen as it provides a visual editor to specify the metamodel for a modeling language, and then is able to generate the GEF, EMF, and Eclipse code required to implement a graphical editor for the specified modeling language. In this way, GEMS reduces the time and effort required to implement the desired graphical modeling tool.

Figure 3.2 shows a screenshot of the DSVL tool, depicted with a complete specification of the European Larch scenario used in this study.

As GEMS is implemented as an Eclipse plugin, the interface is similar to the standard Eclipse interface. The program window, as shown in Fig. 3.2, is split into four main areas. Each of these areas has the following functions:

- The Menu and Toolbars on the top are the same as the standard Eclipse menus and toolbars. In addition to these, as GEMS uses a graphic diagram view rather than the more usual text-based code view, there is an additional toolbar for diagram options including aligning and resizing diagram elements, zooming the diagram, and grid and ruler options.
3.4. METHODOLOGY

Figure 3.1: Baseline tool (LAMOS)

- The Properties Pane on the bottom lists all the attributes of the currently selected element, and allows the user to edit the values of those attributes. The screenshot in Figure 3.2 shows the properties for the entire model, listing a summary overview of all of the elements within the model.
Late Mowing [Disturbance] causes:

- LateMowing-Bromus [Impact]
- LateMowing-Dactylis [Impact]
- LateMowing-Festuca [Impact]
- LateMowing-Sesleria [Impact]
- LateMowing-Larch [Impact]

Impact:
- JuvenileSusceptibilityPercentage: [100.0]
- MatureSusceptibilityPercentage: [100.0]

Attributes:
- Name
- Overview
- Disturbances/Late Mowing
- Impacts/LateMowing-Bromus
- Impacts/LateMowing-Dactylis
- Impacts/LateMowing-Festuca

Disturbance: Enabled: [true], FrequencyOfDisturbanceProneYear: [1.0], MediumToSevereIntensity: [0.0], MildToMediumIntensity: [0.0], SequenceNumber: [0.0]
3.5. **RESULTS AND DISCUSSION**

- The Palette on the right lists all of the possible elements that can be placed inside the model, along with a connection tool to allow users to specify relationships between elements. The groupings or 'drawers' of these elements within the palette are defined when the metamodel is constructed. A new element can be added to the model by dragging the desired element from the palette onto the canvas. The connection tool can be used to join two elements together according to a set of pre-defined rules. These rules restrict which elements can be joined, and what the nature of those joins or relationships can be.

- The Canvas, taking up the remainder of the window, is the working area, where the elements present in the currently opened model are displayed. Elements and connections on the canvas can be rearranged at will.

### 3.4.5 Statistical analysis approach

The Wilcoxon signed-rank test [125] was used to compare the TAM measurement scale items. This test is a paired difference test that assesses whether the population mean ranks of two sample sets differs. This test was chosen as the data from the Likert scale [81] used for each of the measurement scale items is non-continuous and therefore non-parametric. The R software environment [100] was used to perform the statistical analysis.

A two-tailed test was performed in order to test whether the ratings for the DSVL tool differed significantly from those for the Baseline tool. The null hypothesis for each comparison was that the DSVL tool did not receive significantly different ratings compared to the Baseline tool. The results were considered significant for p<0.05.

In all cases, the difference between ratings was calculated by subtracting the rating for the Baseline tool from the same participant’s rating for the DSVL tool. The median of these differences is reported in Tables 3.1, 3.2, and 3.3.

### 3.5 Results and discussion

The results of the two sections of this study are reported and discussed in this section. Firstly, the diagrams produced by participants in Activity 2 (3.4.3) are analysed. Next, the acceptance of the DSVL is compared to the acceptance of the Baseline tool, using the survey which participants completed in Activity 4.
and the semi-structured interview in Activity 5 (Section 3.4.3). Finally, threats to the validity of this study are discussed.

3.5.1 Diagrams

To analyse the diagrams produced in the second activity of this study, the qualitative data analysis technique of coding was used [108]. The contents of the diagrams produced by participants was categorised into having or not having certain qualities and conveying or not conveying particular pieces of information. These categories were identified by analysis of the diagrams produced by participants.

The categories that cover qualities of the diagrams are:

1. Drawing or other sketch, representing the physical landscape layout of the region of interest
2. Landscape plots indicating spatial distribution of species and/or treatments for conceptualisation of the current state or representation of a physical experimental setup
3. Tables, for parameters, results, or any other data
4. Graphs, where elements and relationships between them are represented
5. Links, including arrows, between any pieces of information in the diagram (any graph means that the diagram also includes this feature)
6. Multicoloured, or more than one single colour was used

The categories that cover information are:

1. Parameters of interest, which are varied in the experiment
2. Treatments or land use cases that will be tested
3. Expected results of the experiment
4. Species Elements, representing the plant species as distinct items in the drawing
5. Species Relationships, representing the interactions between the plant species
3.5. RESULTS AND DISCUSSION

Figure 3.3: Part of a diagram drawn by a participant

6. Input data, either as spaces left for unknown input data (as it was not provided in the scenario) or postulated input data

The original diagrams drawn by participants are reproduced in Appendix G. Figure 3.3 shows a diagram that has been extracted from part of a participant’s drawing. This example depicts the ‘drawing or other sketch’ as described above, and was drawn with some variations by four of the participants. The diagram shows a cross section of the physical area of interest in the environmental scenario, and describes what is being tested - the re-establishment of the Larch species into the region that is theoretically suitable for Larch but is currently only grasslands. This kind of diagram was used by participants to help describe and explain the real-world situation described in the scenario.

Another partial diagram is shown in Fig. 3.4. This diagram shows a decision tree for one of the land use cases (early mowing), and explores the effect of the parameters of interest (whether the Larch species has long distance dispersal, whether the Larch species is shade tolerant or intolerant, and whether there are competing grass species present or not) on the final result. This type of diagram, in comparison to those showing a representation of the physical area of interest (Fig. 3.3), shows more abstract concepts and the expected outcomes of the scenario.

Figure 3.5 shows the number of participants’ diagrams that contained each type of quality. Seven of the nine participants included links of some type on their diagram. None of the remaining diagram qualities were commonly included. As mentioned previously, the presence of graphs implies the presence of links of some description.

Figure 3.6 shows the number of participant’s diagrams that conveyed each type of information. As can be seen, all nine participants included the different
CHAPTER 3. SCIENTISTS AND DSVLS

Figure 3.4: Example decision tree graph drawn by a participant
treatments that were described in the scenario. The parameters of interest, input data, and expected results were each included by at least 60% of participants. Species elements were included by half of the participants, but the relationships between the species were only explicitly included in two of the drawings.

The diagrams drawn by participants differed from expectations. The hypothesis was that environmental scientific modelers would draw diagrams that were similar in nature to the diagrams produced by DSVL tools [69, 70, 49, 129], which consist of elements and links in a similar way to the class and dataflow diagrams from the Unified Modeling Language (UML) [102]. However, only four participants drew graphs of some description, which were the closest that any drawing came to the expected diagrams. Only four participants included the plant species as distinct elements in their drawings, and only two of those included some kind of relationship between those plant species indicating that those elements interacted in some way.

Rather than drawing diagrams to explain the components of the environmental scenario and their internal relationships, the diagrams produced by the participants explain the role of the environmental scenario in relation to other scientific knowledge and hypotheses. What is commonly contained within the diagrams
produced by the participants are the treatments to be trialled, the parameters of interest that are to be varied, expected results or the hypotheses to test, and the known input data for the scenario. Each of these categories of information were included by over half of the participants. Collectively, this type of information can be broadly summarised as the context information of the experiment - the information that describes what will be tested and why it is of interest.

Therefore, environmental scientific modelers tend to create diagrams in a different manner and for different purposes to software engineers. When thinking about an experiment, environmental scientific modelers focus on the context information, rather than the specification of the experiment. These results do not support Hypothesis 3.2.

There is no single common manner in which the information is conveyed, with the exception that links or arrows were used to relate parts of a diagram to one another in seven out of nine cases.
3.5. RESULTS AND DISCUSSION

3.5.2 Acceptance of DSVL

Perceived usefulness

The distribution of results for each of the measurement scale items for perceived usefulness are shown in Figures 3.7 and 3.8 for the DSVL tool and the Baseline tool respectively.

Both tools were perceived to be useful as the rating median for all of the measurement scale items were above 4 (Neither), however the ratings for the DSVL tool were slightly higher than those for the Baseline tool.

The comparison of the measurement scale item ratings between the two tools are shown in Table 3.1.

The DSVL tool was significantly (p<.05) perceived to make it easier to perform (Makes Job Easier) and to be more useful (Useful) for environmental modeling compared to the Baseline tool. Although the DSVL tool also received higher ratings than the Baseline tool for the remaining four measurement scale items, none of these were statistically significant. These results indicate that the DSVL tool is perceived to be somewhat more useful than the Baseline tool, but not decisively so.
CHAPTER 3. SCIENTISTS AND DSVLS

Work More Quickly

Job Performance

Increase Productivity

Effectiveness

Makes Job Easier

Useful

Figure 3.8: Perceived usefulness of Baseline tool

Table 3.1: Perceived usefulness - difference between DSVL and Baseline tools

<table>
<thead>
<tr>
<th>Scale Items</th>
<th>p (two-tailed)</th>
<th>median of differences (DSVL-Baseline)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work More Quickly</td>
<td>0.12</td>
<td>1.5</td>
</tr>
<tr>
<td>Job Performance</td>
<td>0.089</td>
<td>1.0</td>
</tr>
<tr>
<td>Increase Productivity</td>
<td>0.10</td>
<td>1.0</td>
</tr>
<tr>
<td>Effectiveness</td>
<td>0.058</td>
<td>1.5</td>
</tr>
<tr>
<td>Makes Job Easier</td>
<td>0.031</td>
<td>2.5</td>
</tr>
<tr>
<td>Useful</td>
<td>0.031</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Perceived ease of use

The distribution of results for each of the measurement scale items for perceived usefulness are shown in Figures 3.9 and 3.10 for the DSVL tool and the Baseline tool respectively.

Both tools were perceived to be easy to use as the rating median for all of the measurement scale items were above 4 (Neither), however the ratings for the
3.5. RESULTS AND DISCUSSION

Figure 3.9: Perceived ease of use of DSVL tool

Figure 3.10: Perceived ease of use of Baseline tool
Table 3.2: Perceived ease of use - difference between DSVL and Baseline tools

<table>
<thead>
<tr>
<th>Scale Items</th>
<th>p (two-tailed)</th>
<th>median of differences (DSVL-Baseline)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to Learn</td>
<td>0.60</td>
<td>0.5</td>
</tr>
<tr>
<td>Controllable</td>
<td>0.14</td>
<td>2.0</td>
</tr>
<tr>
<td>Clear &amp; Understandable</td>
<td>0.18</td>
<td>2.0</td>
</tr>
<tr>
<td>Flexible</td>
<td>0.066</td>
<td>1.0</td>
</tr>
<tr>
<td>Easy to Become Skillful</td>
<td>0.058</td>
<td>1.5</td>
</tr>
<tr>
<td>Easy to Use</td>
<td>0.11</td>
<td>1.0</td>
</tr>
</tbody>
</table>

DSVL tool were slightly higher than those for the Baseline tool.

The comparison of the measurement scale item ratings between the two tools are shown in Table 3.2.

None of the measurement scale items demonstrated a significant difference in perceived ease of use between the DSVL and the Baseline tools, although the DSVL tool received higher results overall.

Overall acceptance

Only two of the six measurement scale items for perceived usefulness show any statistically significant difference between the DSVL and Baseline tools, and none of the six measurement scale items for perceived ease of use show such difference. As perceived usefulness and perceived ease of use are strongly correlated with intention to use and overall acceptance [24], there is little evidence to support the DSVL tool being more acceptable than the Baseline tool (Hypothesis 3.1).

Perceived modeling utility

The distribution of results for each of the measurement scale items for perceived modeling utility are shown in Fig. 3.11 and Fig. 3.12 for the DSVL tool and Baseline tool respectively.

Both tools were perceived to have modeling utility as the rating median for all of the measurement scale items were above 4 (Neither), however the ratings for the DSVL tool were higher than those for the Baseline tool.

The comparison of the measurement scale item ratings between the two tools are shown in Table 3.3.
3.5. RESULTS AND DISCUSSION

Easily Understand Scenario
Clear View of Scenario
Explore Scenario Easily
Grasp Scenario Structure Easily
Easy to Communicate Scenario
Understand Different Scenario

Figure 3.11: Perceived modeling utility of DSVL tool

Easily Understand Scenario
Clear View of Scenario
Explore Scenario Easily
Grasp Scenario Structure Easily
Easy to Communicate Scenario
Understand Different Scenario

Figure 3.12: Perceived modeling utility of Baseline tool
Table 3.3: Perceived modeling utility - difference between DSVL and Baseline tools

<table>
<thead>
<tr>
<th>Scale Items</th>
<th>p (two-tailed)</th>
<th>median of differences (DSVL-Baseline)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easily Understand Scenario</td>
<td>0.020</td>
<td>2.0</td>
</tr>
<tr>
<td>Clear View of Scenario</td>
<td>0.014</td>
<td>2.4</td>
</tr>
<tr>
<td>Explore Scenario Easily</td>
<td>0.013</td>
<td>2.0</td>
</tr>
<tr>
<td>Grasp Scenario Structure Easily</td>
<td>0.0083</td>
<td>2.0</td>
</tr>
<tr>
<td>Easy to Communicate Scenario</td>
<td>0.0084</td>
<td>2.5</td>
</tr>
<tr>
<td>Understand Different Scenario</td>
<td>0.0088</td>
<td>2.5</td>
</tr>
</tbody>
</table>

All of the six measurement scale items for perceived modeling utility showed a significant (p<.05) difference in the ratings between the DSVL and the Baseline tools, with the DSVL tool consistently rated higher than the Baseline tool. Therefore, the DSVL tool is perceived to be more useful for modeling than the Baseline tool, which supports Hypothesis 3.3.

Qualitative comments

The qualitative comments collected during the structured interview stage of this study support the results of the survey questionnaire. In general, comments regarding the DSVL were mostly positive, with all nine participants making at least one positive comment and only four making one or more negative comments regarding the DSVL tool. For the Baseline tool, positive comments were recorded from six participants, and negative comments also were recorded from (a different but overlapping set of) six participants. All of the comments for each of the tools are summarised below, with the number in parentheses indicating how many participants made the comment.

The DSVL tool received the following positive comments:

- Better overall view of the experiment, making elements and relationships clear rather than having to have an internal mental image of the experiment (8)
- Clear and visible options with more immediate feedback (4)
- Easier to explain the scenario to others (5)
3.5. RESULTS AND DISCUSSION

• Easy to use (2)
• Easy to learn (2)
• Easy to find particular parts of the model or particular parameters (1)

The DSVL tool received the following negative comments:

• Difficult to learn the new style of interface that differs from most computer software programs (3)
• Parameter input is clunky, having to switch between elements and parameters rather than just having a list/table that can be easily filled out (1)

The Baseline tool received the following positive comments:

• Quicker for just inputting data and parameters (3)
• Easier to learn/use as it is in the same style as most computer software programs (3)

The Baseline tool received the following negative comments:

• Difficult to know at any one time where the section that is currently being worked on fits into the big picture (2)
• Difficult to learn and non-intuitive (2)
• More difficult to find particular elements or parameters (2)
• Does not assist in thinking about hypotheses or experimental design (1)

These comments support the quantitative survey results. In particular, in the area of viewing and exploring the scenario and experimental setup, the DSVL tool received positive comments, whereas the Baseline tool received negative comments. In regard to ease of use and usefulness, both tools received mixed positive and negative comments.

3.5.3 Threats to validity

The small sample size of this study limits its external validity. In addition, there is some geographical selection bias, as all participants were solicited from a single university, the ANU. However, as the ANU is a world-class institution, and its
environmental sciences area has ties to other universities both nationally and internationally. Environmental scientists at the ANU could be considered to be representative of environmental science researchers overall.

Three of the nine participants in this study had not been involved with producing software code after their university-level education. However, all were involved in the development and use of environmental models. These tasks involve the setup and specification of environmental models, which was the focus of the DSVL and Form tools tested in this study.

Particular quirks or other elements of each individual tool may have had an impact upon the results, particularly for perceived ease of use. To mitigate this, participants were walked through the usage of both tools, in order to ensure that both tools were provided equal treatment and participants did not get stuck at any point. Both tools had their own problems and neither was a complete, polished product.

The walkthrough of both tools was provided by a single researcher, introducing an element of experimenter bias into the study. Care was taken to explain the two tools in a dispassionate manner, with instructions restricted to the order of tasks and the location of the relevant interface elements.

3.5.4 Summary of results compared to hypotheses

A summary of the results relevant to the hypotheses of this study (Section 3.3) are presented below:

**Hypothesis 3.1.** The DSVL tool is more acceptable than a baseline tool.

Hypothesis 3.1 is not supported by this study, as there was no significant difference the perceived usefulness or perceived ease of use between the DSVL and Baseline tools, and therefore according to the TAM [24] no difference in their acceptance.

**Hypothesis 3.2.** The diagrams drawn by environmental scientific modelers to explain their experimental scenarios are similar to those of software engineers.

Hypothesis 3.2 is also not supported by this study, as the diagrams drawn environmental scientific modelers differ in both form and function from diagrams commonly drawn by software engineers.

**Hypothesis 3.3.** Environmental scientific modelers more easily understand and communicate the specified experimental scenario using the DSVL tool, compared with a baseline tool.
3.6. IMPLICATIONS

Hypothesis 3.3 is supported by the results of this study. Environmental scientific modelers rated the DSVL tool significantly better in regard to perceived modeling utility compared to the Baseline tool, and the qualitative comments gathered also support this result.

3.6 Implications

Based on the diagrams that were drawn by participants, the elements that are considered by environmental scientific modelers when describing an environmental scenario differ from those that were hypothesised. The diagrams created did not focus on the specification and setup of the scenario and the relationships between elements within the scenario. Instead, the diagrams contained information on the treatments, parameters, expected results, and input data - the context information for the experiment that explains what will be tested and why it is of interest. These types of information were not able to be represented in either of the tools tested in this study. No other DSVLs in environmental science fields capture this information, instead concentrating on describing and modeling the scenario details required for simulation [12, 36, 91]. It can be hypothesised that a tool that supported recording of such contextual information would have improved usefulness compared to existing tools lacking this ability.

The survey results, backed up by the qualitative comments, show that there is no significant difference between the DSVL and Baseline tools in regard to perceived usefulness or perceived ease of use. Therefore, their acceptability by environmental scientific modelers does not differ significantly. Although DSVLs have been shown to provide many benefits relevant to environmental scientific modelers, these benefits are not reflected in improved perceived usefulness. Similarly, although DSVLs have been postulated to provide accessibility and usability benefits for domain experts, these are not reflected in improved perceived ease of use. This lack of difference in acceptability implies that environmental scientific modelers would not be more likely to use either tool over the other. One potential reason for this could be because scientists are more concerned about the science and context behind their experiment and, barring significant problems, the specific tool which is used does not make a difference.

The DSVL tool was perceived to be significantly more useful for understanding, exploring, and communicating a scenario. That these results differ from
perceived usefulness may imply that environmental scientific modelers consider understanding, exploring, and communicating scenarios as different activities to performing environmental modeling in general.

3.7 Conclusion

In summary, the important conclusions that can be drawn from this study are as follows:

1. Environmental scientific modelers do not perceive the DSVL to be significantly more or less acceptable compared to the Baseline, non-DSVL tool.

2. Environmental scientific modelers focus on the context information of an experiment, and specifically the parameters of interest, input data, and expected results, when explaining an environmental scenario.

3. Environmental scientific modelers perceived the DSVL to be significantly better compared to the Baseline, non-DSVL tools for the purposes of understanding, exploring, and communicating an experiment.

4. Environmental scientific modelers consider understanding, exploring, and communicating an experiment as distinct from environmental modeling in general.

The potential usefulness of DSVLs in scientific software development is not mirrored by an improved acceptance of DSVLs by scientists. The DSVL trialled in this study was not perceived by environmental scientific modelers to be any more (or less) useful or easier to use compared to another existing non-DSVL tool, and therefore, according to the TAM, the DSVL is not more (or less) acceptable to environmental scientific modelers. This may be as scientists concentrate on the context behind the experiment, and little importance is placed on the specific tool used to specify and implement a software simulation of the experiment.

However, the DSVL is considered significantly better for understanding, exploring, and communicating an experiment. This indicates that DSVLs have perceptible benefits to environmental scientific modelers for purposes other than specifying and implementing an experiment. This use of DSVLs for purposes other than specification and implementation in the area of scientific research presents an opportunity to apply DSVLs for reasons and purposes not previously explored in software engineering.
3.7. CONCLUSION

A broader view of the roles which software engineering currently does and potentially could play in scientific research can be taken. This systems view of scientific research and software engineering allows analysis of the current interaction between the two fields, and allows identification of other areas where software engineering tools, techniques, and methodologies could be applied to benefit scientific research. In this way, a new application of a specific software engineering tool, such as DSVLs, within the field of scientific research can be both clearly identified and justified.
Chapter 4

Scientists, Software, and Systems

4.1 Introduction

This chapter explores the broader implications of the research presented in this thesis thus far. The results are discussed from a systems viewpoint, and how systems thinking and systems engineering can be applied to scientific research and scientific software. This exploration culminates in a number of insights into the development and use of software in scientific fields.

First, the major concepts of systems, software systems, systems engineering, software engineering, and science are explored. Second, the systems approach is applied to scientific research and software engineering, and the holistic view of scientists towards scientific research and the more focused view of software engineers towards software development are compared and contrasted. Third, a systems approach to scientific software development is discussed, and an argument made for software engineers to broaden their perspective and look for ways in which software engineering can aid scientific research as a whole, rather than simply improving scientific software development in isolation. Finally, the use of DSVLs to aid scientists in managing the context information of experiments is proposed, as one potential application of software engineering to aid the system of scientific research.

4.2 Background

This section explores the meaning of the terms ‘system’, ‘software system’, ‘systems engineering’, ‘software engineering’, and ‘science’, in order to provide a basis for the subsequent discussion.
4.2.1 Systems

There are many different definitions of systems. At the broadest level, Jordan [65] defines the only requirements of a system to be a set of identifiable entities with identifiable connections between them. A more specific definition is provided by Kleiner [114], where a ‘system is a perceived whole whose elements “hang together” because they continually affect each other over time and operate toward a common purpose’.

Ackoff [1] provides a more detailed description, which is claimed to capture the core of agreement between various definitions. Ackoff’s description defines a system as a whole that consists of at least two component parts, which satisfies the following five conditions:

1. The system as a whole must have a defining property or function. This is the purpose of the system, as referred to by Kleiner [114]. The defining properties, functions, or purposes of the system are the reasons for the system’s existence. For example, a school is a system which provides education for children, an airplane is a system which provides transportation of people and/or goods.

2. Each component part of the system can affect the behaviour or properties of the system as a whole. Changing the behaviour of an individual part of the system can change the behaviour or properties of the system as a whole.

3. There are essential parts of a system - namely, a subset of the component parts of a system where the removal of any one of those parts results in the system being unable to satisfy its defining properties, functions, or purposes.

4. No essential part of a system works independently, and the effect of each essential part of a system depends on at least one other essential part of the system. This means that essential parts must interact with other essential parts in the system.

5. No subset of essential parts of a system works independently, and the effect of any subset of essential parts of a system depends on at least one essential part of the system not in the subset. This means that all of the essential parts of a system are connected, and cannot be split into two independent subsets of essential parts.

This can be summarised and oversimplified to the following: ‘A system is a whole that cannot be divided into independent parts without loss of its essential
4.2. BACKGROUND

properties or functions[1]. This definition of a system encompasses many situations, organisations, and constructions in the world. The classic example of a system is a car. The primary purpose of a car is to transport people across land. Without any one of its essential components - the engine, wheels, steering, and brakes - a car cannot operate for its primary purpose. Each of those components is dependent upon at least one of the others.

Attempting to change and optimise a single part of a system in isolation may result in unintended and detrimental consequences on the system as a whole, due to the dependencies and relationships between the parts. Using the car example, one way the engine may be improved may be to improve its power. However, without more robust wheels, the wheels may slide instead of provide traction, or fail structurally due to the increased forces. With a more powerful engine and a higher top speed or cargo capacity, stronger brakes may be required to stop the car in an acceptable space. Without considering the car from a systems perspective, a naive optimisation of one particular part can have detrimental effects on the whole. Another example of naive optimisations resulting in unintended consequences within a system may be improvements in software engineering within science, which will explored in Section 4.4.

The two important aspects of a system from this definition are that systems exist for a particular purpose, and a system’s component parts cannot be separated without impacting the system’s purpose. The purpose of a system may vary depending upon the perspective being considered, and changes in perspective and purpose may result in changes to what is considered part of the system and what the essential components of the system are.

4.2.2 Software systems

Software systems have traditionally been defined as not only the source code of a computer program, but also include other outputs of the software development life cycle such as requirements, design, specification, tests, test results, and documentation [48]. This definition describes a software system from a perspective of software construction - namely, what components are involved in the development and creation of a computer program.

However, an alternative view from a systems perspective would be that a software system is a system that includes computer software or computer programs. This systems view of a software system revolves around the overall system, that the computer program operates within. The overall system - the software system
- includes both the computer program and all of the other required components to perform the system's primary purpose. This may also include the electronic hardware, any mechanical hardware, users, and administrators. As an example, the software system of an automatic teller machine (ATM) includes the computer program which runs on the ATM, as well as the computer hardware, the mechanisms in the ATM, the banking system which the ATM is linked to, and the users of the ATM. This entire system of banking access can be viewed as a software system - a system which involves the computer program of an ATM.

This alternative view of software systems is not commonly considered by software engineers, and this issue will be explored further in Section 4.3.2.

4.2.3 Systems engineering

Systems engineering applies the systems and systems thinking approaches to the design and management of engineering projects. Systems engineering has evolved to become a discipline in and of itself, used to manage complex projects with many interacting components across multiple disciplines.

Systems thinking [19] is an approach to understanding how things work by considering them as parts of an interrelated whole, or in other words as part of a system (as defined previously). Instead of viewing situations as simple cause-and-effect relationships, or reducing situations to unrelated components which are analysed and explained in isolation, the systems thinking approach embraces the nature of systems as being comprised of many interrelated components. Systems thinking looks at the wider environment of a situation, and analyses how components of a system are related to both one another as well as other systems.

In essence, systems thinking focuses on the role of an object within its wider system. This is contrasted to reductionist thinking [64] where the focus is on determining how an object works, and that the functions of the object can be explained by breaking it down into smaller parts. Reductionist thinking explains how an object works by understanding these smaller components individually and in isolation from one another, and that the individual behaviours of these components are added together to explain the entirety of the original object's behaviour.

This systems approach retains and focuses on the relationships and behaviours which occur in the relationships between the components of a system. Systems thinking allows the analysis of how the system as a whole would react to a change in a single component, by understanding the relationships between that compo-
4.2. BACKGROUND

...system and others within the system. Systems thinking also takes into account the overall purpose of the system, which may vary depending on what perspective the system is viewed from.

4.2.4 Software engineering

Software engineering is the application of engineering principles to software products. In theory, this means the application of knowledge of various types to the design, development, and maintenance of software products. This revolves around systematic approaches to software development. Software engineering has resulted in numerous tools and methodologies to support various stages and activities of the software development process. Software engineering has traditionally focused on the development of software as an isolated product [117]. This matches with the traditional definition of a software system as described in Section 4.2.2, which focuses on how the software system works.

When a systems perspective is taken of software, it can be seen that software forms a single component of larger systems. This viewpoint focuses on the role of the software system (as defined in Section 4.2.2) within the larger system. Therefore, traditional software engineering is performed on an isolated component of larger systems. As explored in Section 4.2.1, due to the relationships and dependencies between parts of systems, optimisation and improvements of a single part may have unintended consequences and be detrimental to the system as a whole.

There has been some discussion of the role of software within the context of a broader system [116], bringing a systems approach to software engineering as per the alternative view of software systems (Section 4.2.2). The ANU in particular began to offer a course in systems engineering for software engineers in 2012 [38]. However, software engineering tools, techniques, and methodologies have not yet caught up with this systems approach.

4.2.5 Science

Science is the discipline of improving understanding about the world. Many scientists are aware of and seek to understand the complex systems which exist in the world. Systems science is a particular scientific field which covers complex systems of all types, particularly those which cover multiple disciplines.

The scientific method forms the basis for scientific research. The scientific
method involves a number of activities, which are described below. Although these activities can be described in a sequential fashion, scientific research is much more complex than a simple sequence of activities. The scientific method involves the following activities [45]:

Develop Questions is the process of determining gaps or unknowns in existing scientific knowledge and framing research questions to address those gaps and unknowns.

Develop Hypotheses involves coming up with potential answers to the previously posed research questions.

Test Hypotheses is the process of determining whether the world behaves as predicted by the hypotheses, through observations and experiments.

Analyse Results involves determining what the results or output of the experiments performed actually mean, particularly in terms of the hypotheses posited and the questions posed.

Although systems approaches have been used in scientific research since at least the 1950s [34], there has been little or no analysis of scientific research itself from a systems perspective. If a systems view is taken of scientific research itself, the overall goal of the system of scientific research is to increase understanding of the world. If the above activities in scientific research are organised into a system which includes the main relationships between the activities, it may appear as shown in Figure 4.1. In this diagram, the collections of scientific knowledge and engineering expertise have been added, and the ‘Test Hypotheses’ activity has been expanded into the separate stages of designing, building, and running experiments. These additions and expansions are defined as follows:

Scientific Knowledge is the body of existing scientific knowledge.

Engineering Expertise is the technical knowledge required to build the required equipment for an experiment.

Design Experiments is the conception of scientific experiments, which are activities which test whether observations of the world support or falsify predictions made by the hypotheses. The design of such experiments should take into account potential risks, errors, and biases.
4.2. BACKGROUND

Figure 4.1: The system of scientific research
Build Experiments is the process of setting up the designed experiments. Depending on the exact nature of the activity, the amount of effort done in this stage varies, and can include gathering subjects, acquiring or constructing measurement equipment, and/or building software to run a simulation.

Run Experiments is the execution of the experiments.

Although Figure 4.1 shows scientific research as a mostly linear process from developing questions through to analysis of results, in reality there are many interactions between successive steps. An example can be that building or running an experiment may indicate flaws in the design, which result in changes to the design of the experiment even before results have been collected or analysed. These feedback loops have been omitted from Figure 4.1 as they do not play a significant role in the subsequent discussion regarding the application of software engineering within the system of scientific research (Sections 4.3 and 4.4).

Along with the above process, there is increasing attention being paid to data-first or data-intensive science, a process which has also been labelled the 'fourth paradigm' [59] (as opposed to the first three paradigms of experimental science, theoretical science, and computational science [58]). In data-intensive science, rather than designing, building, and running new experiments, existing sets of data are analysed in order to gain new insights. This approach is enabled by new technologies which allow scientists to collect, manage, analyse, and display data and relationships between data in new ways.

4.3 From a systems thinking point of view

This section presents thoughts about the activities of scientific research, software engineering, and the differences and interactions between the two sets of activities, all from a broader systems thinking perspective.

4.3.1 Thinking about scientific research

As seen by the previous experiment (Chapter 3), scientists focus on the context information of a scientific experiment. This context information consists of information such as the parameters of interest, input data, and hypotheses of the experiment.

Scientists perform scientific experiments to improve their understanding and knowledge. Therefore, from the point of view of systems thinking, scientific ex-
periments are a solution to a problem - the 'problem' being insufficient knowledge about a scientific domain. This 'problem' may arise due to increased scientific understanding being required to create or improve something, or due to simple curiosity and a desire to better understand the world. The scientific experiment is performed within the system of scientific knowledge - a large, interconnected collection of individual pieces of understanding and knowledge.

Context information relates the scientific experiment to the system of scientific knowledge. This relationship defines the purpose and meaning of the environmental experiment. It also provides the background or previously existing information which is fed into the experiment, and also defines the experiment's scope and applicability. Without the context information, the role of the scientific experiment within the system of scientific knowledge is unknown.

The fact that scientists focus on context information when describing a scientific experiment (shown by the experiment presented in Chapter 3) indicates that they are aware of the importance of the relationships between the experiment and the system of scientific knowledge. The focus on this relationship is evidence that scientists consider scientific research from a systems perspective. Scientists acknowledge that without context information, much of the meaning behind the experiment is lost. Without the parameters of interest, what is being tested by the experiment is unknown. Without the input data, the previous knowledge used as a basis for the experiment is unknown. Without the hypotheses, the anticipated results and the reasons for those anticipated results are unknown.

### 4.3.2 Thinking about software engineering

Scientists focus on the relationships between the scientific experiment and the system of scientific knowledge. In comparison, software engineers often focus on the software being produced as an independent entity to the system in which the software will be deployed.

Software engineering tools, techniques, and methodologies focus on improving the process of software construction. Examples of such methodologies include DSVLs, requirements engineering [118], agile software development [84], and Model-Driven Engineering (MDE) [107]. In the scientific software field, numerous frameworks, workflow management systems, and simulation platforms have been produced. All of these software tools and software engineering methodologies address the solution space - that is, aiding the development of a software system once the problem has been identified and a solution decided upon. Agile
development processes and principles such as those developed by De Roure and Goble [29] focus on faster iterations through the software development process, producing incremental and iterative changes quickly, with the intent that these smaller changes more closely address the requirements of users and provide solutions or improvements in a more timely manner. Sometimes, these tools and methodologies are less widely used than otherwise hoped or expected.

In general, it appears as if software engineering tends not to take a systems thinking approach. Instead, software engineering tools, techniques, and methodologies focus on how the software works - improving the manner in which software is constructed or the quality of the end result of the software alone. These goals are rarely placed in context of the overall system in which the software will operate - instead, the efficient production of high-quality software is viewed as end result of software engineering.

4.3.3 Differences between scientific research and software engineering

Scientists develop and use software regularly [51], and produce large programs which are successfully used for scientific research purposes. However, that there remains a lack of use of methodologies commonly considered basic by the software engineering community, although the situation has much improved in recent years, as shown by the survey in Chapter 2.

The study in Chapter 3 shows that environmental scientists focus on the context information of an environmental scenario, rather than the details of implementation or what tool is used to implement it. This supports previous literature, which has described the focus of scientists as performing science, not developing code [7, 112]. This focus on the science rather than the software is at odds with the focus of software engineering in the scientific field so far, and more widely the overall focus of software engineering in general.

This difference in focus may be the reason why there remains a lack of adoption of software engineering practices and methodologies in scientific software engineering. Research and literature from software engineering persist in describing best practices for scientific computing [126]. Software engineering can improve scientific research via improvements in scientific software. Direct education of scientific software developers has led to the successful adoption of software engineering tools, techniques, and methodologies [5]. However, there remains an overall lack of adoption of such practices in scientific software development (Sec-
4.4 A systems thinking approach to scientific software

Scientific software development, in a similar manner to software engineering in general, has focussed on the introduction of practices and methodologies which improve the process of software construction. More specifically, the use of tools to support development, improved testing procedures, and documentation have been identified as areas to improve [126]. The arguments for improving the use of software engineering in scientific software development are the same as the arguments for improving software development in general - improved developer productivity and improved quality of the resultant software.

However, scientific software forms only a part of the system of scientific research. The goal of the system of scientific research is not improved scientific software developer productivity, nor improved quality of scientific software. The goal of the system of scientific research is to increase understanding of the world, as identified in Section 4.2.5. As explored in Section 4.2.1, attempting to change or optimise a single component part of a system may have detrimental effects on the whole. Therefore, software engineers should not try to improve scientific software development for the sake of improved scientific software developer productivity and quality of scientific software. The goal of software engineering in scientific research should be to increase scientific research productivity in order to increase understanding of the world.

An influence diagram showing the relationships between scientific research activities and software engineering tools, techniques and methodologies is shown in Figure 4.2. Although the activities of scientific research are generally applicable to all types of scientific research, the influence of software engineering tools, techniques, and methodologies is mostly limited to scientific research activities in the area of scientific computing. The diagram shows a selection of software engineering tools, techniques, and methodologies which have been and are used to support scientific research, however as indicated by the ellipsis there are many more tools, techniques, and methodologies available.

It can be seen from Figure 4.2 that the existing applications of software engineering tools, techniques and methodologies in support of scientific computing are focused on the latter half of scientific research activities - namely, experimental
Figure 4.2: Influence diagram of scientific research activities (in the area of scientific computing) and software engineering tools, techniques, and methodologies.
4.5 DSVLS and context for scientific experiments

In Chapter 3, it was identified that the demonstrated benefits of DSVLS match well with the ongoing issues and areas considered important in scientific software development. However, the DSVL tool investigated was not found to be more accepted by scientists, compared to an existing baseline tool. As explored above,
extending the DSVL tool to allow its application to other activities of scientific research may increase its acceptance by scientists.

The context information of a scientific experiment describes the relationship that experiment has with the broader system of scientific knowledge. As also identified in the experiment presented in Chapter 3, this information is of importance to scientists when explaining scientific experiments. Software engineering may be able to provide ways in which the treatment context information can be improved.

If context information recording could be supported with a tool or methodology, then potentially it would be more valuable and more accepted by scientists as it matches their area of focus. If scientists use the tool or methodology because it aids them in the capture and recording of context information, then it would be easier to convince them to also use the same tool for building scientific software (assuming it also provides this functionality).

A DSVL could be used to record context information by providing a graphical management interface, allowing such information to be embedded alongside the software system being specified. This would allow scientists to record the relationships between particular items of contextual information and specific parts of the associated scientific software program.

Current approaches for recording context information by scientists are explored in greater detail in the next chapter (Section 5.2).

4.6 Conclusion

A systems thinking approach reveals the narrow focus of software engineering on tools and methodologies which directly support software construction. Software engineering does not focus on the roles which software plays within the systems of which software is a part. An example of this is the interaction of software engineering and scientific software development. In light of this difference, other ways in which software engineering tools, techniques, and methodologies could be of assistance to scientific research can be explored.

One potential application is using DSVLs to allow scientists to record context information regarding their experiments. This is hypothesised to allow them to record this information easily and alongside the specification of a software simulation for an experiment. If scientists are willing to use such a tool due to its functionality for recording context information, this may improve its over-
all acceptability and also encourage the scientists to also use it to specify their software.
Chapter 5

DSVLs, Scientists, Context, and Provenance

5.1 Introduction

Software engineering has thus far focused mostly on tools, methodologies, and processes for directly supporting software construction, rather than on the role of software to support and improve systems. The discussion of scientists, software, and systems presented in Chapter 4 culminated with the following question: ‘Are scientists willing to use software engineering ideas to support scientific research activities other than developing scientific software?’ In order to address this question, a study of one such example of using a software engineering idea for supporting a non-scientific software development activity was performed.

Based on the study presented in Chapter 3, the use of DSVLs to support the recording of context information was identified as an area of further interest. This use of DSVLs would be a new role of a software engineering idea in the system of scientific research, outside of developing scientific software. The study determined that scientists focused on the context information when explaining a scientific experiment. Context information explains what is being tested in the scientific experiment, and also why it is of interest and therefore why the scientific experiment is being performed. Current DSLs and DSVLs for environmental science focus on the specification of an experimental scenario [12, 36, 91].

Little research has been performed regarding the capture and use of context information, although there has been some attempt to merge record context or semantics alongside provenance information [103]. Although the exact nature of provenance varies between communities, provenance is used for assessing the au-
thenticity and integrity of experiments [20]. In this way, provenance information encompasses the source of any data, such as who produced the data, when it was produced, where the data is relevant to, and how it was collected or generated. Several systems have been developed and published to record this type of information in scientific research [4, 40, 46, 62, 88, 97, 128]. However, there has been little analysis of the acceptability of such systems.

The terms curation and metadata often are mentioned in conjunction with provenance. Curation is the activity of managing data and promoting its use by ensuring it accessible and fit for purpose [82]. Curation can also involve maintaining links between the data and other materials. In this way, the act of curation involves the production, maintenance, and distribution of provenance information. Metadata has been defined as data which supports discovering, understanding, and managing other data [28]. However, others define metadata more broadly, using the terms metadata and provenance interchangeably [115]. In order to avoid confusion, the use of the term metadata will be avoided.

This chapter presents an empirical study of an extended DSVL tool for environmental modeling, which allows users to also record context and provenance information about the environmental experiment being specified. The acceptance of the extended DSVL tool was measured, to determine whether this extended DSVL tool showed any differences compared to the original DSVL or the Baseline tools which were analysed in Chapter 3. As little was known regarding the current practices of scientists for recording context and provenance information, semi-structured interviews were conducted to gain additional understanding.

The scope of this study was limited to environmental modeling for a number of reasons. Primarily, in order to be able to compare the results of this study with those from Chapter 3, similar sample characteristics were required. In addition, the nature of the DSVL being tested would remain constant, with only the addition of the ability to record context and provenance information. The reasons presented in Chapter 3 also remain, those being to narrow the focus to a clearly defined subset of scientists rather than attempting to cover the wide variety of scientific research fields; the existence of previously-developed DSLs; and that modeling in general forms a significant part of scientific research [43].

First, existing literature regarding the capture of context and provenance information is summarised. Second, the research questions and hypotheses for this study are detailed. Third, the methodology used in this study is described. Fourth, the results of this study are presented and discussed. Finally, the implications of this study are explored.
5.2 Existing literature

Although most, if not all, scientific papers present the relevant background information and reasons for the research being published, little research has been performed regarding how this type of information is recorded. The laboratory notebook is a traditional recording tool for all scientists [66], however there has been little study of its actual use and usefulness.

Dealing with and recording data provenance information in scientific research has received a significant amount of attention. Literature exists regarding the nature of provenance information as well as for systems to support the recording of this information, which will be explored below. However, there is little research into the current practices for provenance recording, or adoption of provenance recording techniques. It has been asserted that there are few people actually producing provenance information [18], and that a number of basic problems with provenance in today's systems exist [20]. These basic problems are incomplete, unreliable, and insecure provenance data, which is heterogeneous and non-portable. However, there is little evidence provided to support these claims.

A taxonomy of data provenance characteristics has been developed by Simmhan et al. [115]. This taxonomy covers the use, subject (data or process oriented and granularity), representation, storage, and dissemination of provenance information. This taxonomy is applied to seven current projects in data provenance research and two historical projects, and some of these projects are presented later in this section. Although this research represents a comprehensive survey of techniques being used in respect to the functions of a provenance system, there is no analysis of the acceptance of any of the provenance systems that were analysed.

There exist many systems and architectures for recording provenance information in scientific fields, some of which are covered by Simmhan et al. [115]. The literature regarding these individual systems concentrates on the construction and the features of each system being covered, with the benefits that the use of these systems would bring being of secondary importance. Similar to Simmhan et al. [115], little to no analysis is performed on the acceptability of such systems to potential users. Some of the systems which have been developed for recording provenance information are summarised here.

Chimera [40] is a prototype system which records data and provenance information in an integrated community environment. Chimera was developed for scientists in areas of physical sciences, such as the fields of high energy physics and astronomy. Chimera is asserted to be able to improve the capture, discovery,
use and management of data and derivative data. Initial results from successive 
work suggest that at least some of the benefits claimed for virtual data systems 
can be realised in practice [41]. No analysis of the acceptability of Chimera is 
performed.

The Collaboratory for the Multi-scale Chemical Sciences (CMCS) [97] is a 
toolkit for collaboration and data management, which also manages provenance 
and workflow information. The tool is intended to enable the sharing of such 
information between differing scientific domains which are involved in Chemical 
Sciences. The usefulness of the project in enabling users to share data libraries 
and assess the quality of said data using provenance information is briefly men­
tioned.

The Kepler scientific workflow system [4] allows scientists to design and exe­
cute workflows in a workflow environment. Kepler is designed to support multi­
disciplinary and multi-project workflows, and records the context of the exper­
iment, the input data, metadata, and workflow information. Importance was 
placed on the ease of use and efficiency of the Kepler system during its devel­
opment, minimising the performance overhead which Kepler imposes. Kepler is 
asserted to be useful and easy to use, however no analysis of this is per­
formed.

The Provenance-Aware Service-Oriented Architecture (PASOA) [88] is a pro­
totypte tool for recording provenance. The requirements for the architecture was 
developed from a series of use cases from a diverse range of scientific fields, how­
ever it was concluded there was little which is common across all of the use cases. 
The approach used to develop PASOA is compared to other provenance recording 
approaches, which were seen as more specific and therefore did not cover all of 
the use cases which were have identified. No analysis of the acceptability of the 
PASOA prototype is performed.

Taverna [62], one of the software products produced by the my Grid proj­
ect [128], is a workflow management workbench, allowing users to write, automate, 
and share bioinformatics analyses. It has reportedly had initial success, with 
scientists spending less time performing such analyses. Although there is no 
formal analysis of acceptability of the system, the issue is raised through the 
questions of how provenance data could be best presented to users, the usability 
of the system, and the scalability of the handling and presentation of provenance 
information.

Velo is a knowledge-management platform for modeling and simulation [46], 
providing a collaboration environment and content management system. Velo 
also provides a tool integration framework to allow users to invoke and record the
use of tools. Velo is used in a number of areas, all of which involve the developers in some way. The features, extensibility, and applicability in various domains of Velo is described. However, the acceptability of the system is not analysed.

Finally, a system for automatically capturing ‘the detailed context of experiments’ is presented by Davison [27]. The use of the term context in this paper is more specific compared to its usage within this thesis. In this paper, context information refers to the metadata required to replicate the computational results - namely, the hardware platform, operating system, source code, compilation process and compiler version, input data, and output data. The sample implementation, Sumatra, provides a command-line interface to run a computation and collect input, output, software, and hardware information. A web browser interface is also mentioned, but not described in detail nor is available at present. The potential benefits of improved reproducibility of experiments by automating the capture of metadata information are explored, however no analysis of the acceptability of the proposed system is performed.

5.3 Goals and hypothesis

Qualitative data from the previous study (Chapter 3) shows that when asked to describe a scenario, environmental scientific modelers concentrate on the context information for the environmental scenario - in other words, the known data, decision points or parameter variations, and expected outcomes. In comparison, DSVLs currently used in environmental modelling [12, 36, 91] concentrate on describing and modeling the scenario details required for simulation. The DSVL presented in the previous study also concentrates on this area.

Based on the results of the previous study, it was hypothesised that adding the ability to record context information to the DSVL would improve the DSVL’s acceptance by environmental scientific modelers. Therefore, the following research questions were posed:

1. Would environmental scientific modelers be willing to use a DSVL for recording context and provenance information?

2. Would a DSVL which allowed environmental scientific modelers to record context and provenance information be more acceptable to environmental scientific modelers than a DSVL which did not allow recording of such information?
However, as little was known about how environmental scientific modelers currently capture context and provenance information, further investigation into how these types of information are currently recorded was also required. In order to address this, the following research question was added:

3. How do environmental scientific modelers currently think about and record context and provenance information?

The hypotheses for each of research questions are as follows:

**Hypothesis 5.1.** A DSVL which allows recording of context and provenance information would be accepted by environmental scientific modelers.

**Hypothesis 5.2.** A DSVL which allows recording of context and provenance information would be more acceptable to environmental scientific modelers than a DSVL which does not.

**Hypothesis 5.3.** Environmental scientific modelers currently consider and record context and provenance information in an informal and ad-hoc manner, and systematic recording of context and provenance information is not performed.

### 5.4 Methodology

This section describes the experimental conduct, details regarding participants, the activities which the participants performed, the tools compared, and the approach used to perform statistical analysis on the data collected.

This study was performed under a variation of the ethics approval for the study presented in Chapter 3. The ethics approval form for the original study is included in Appendix C.

#### 5.4.1 Experimental conduct

The experiment consisted of one-on-one sessions, each of which was scheduled to last approximately thirty minutes. Some sessions took longer amounts of time due to extended discussions or semi-structured interview segments. In each session, participants were asked to perform a number of tasks, which are described in Section 5.4.3. Sessions were conducted during February to April in 2012.

Potential participants were contacted by email directly and through email mailing lists.
5.4. METHODOLOGY

5.4.2 Participants

Participants were elicited across various Australian universities. Participants were required to be involved in environmental modeling. No specific requirements were made regarding the exact nature of that involvement, such as the field of work (for example: hydrology, ecology, atmospheric) or type of modeling (for example: statistical, computational simulation).

Seventeen environmental scientific modelers participated in this study. These participants were different from those who participated in the study presented in Chapter 3, in order to avoid bias in the result due to previous experience with the DSVL tool. Fourteen of these were academics, with the remainder consisting of two postgraduate students and one other researcher. The participants had varying levels of previous modeling experience as shown in Figure 5.1, with the majority (ten) having 11-20 years of experience.

The information sheet used for this study is included in Appendix I. The consent form, which all participants were required to complete prior to participation, is included in Appendix J.

Figure 5.1: Modeling Experience
5.4.3 Sessions and activities

Each session consisted of the following five activities:

1. First semi-structured interview, regarding the participant’s work, context, and provenance
2. Guided walkthrough of the extended DSVL tool
3. Survey of acceptability of the extended DSVL tool
4. Second semi-structured interview, regarding the participant’s opinions and comments regarding the extended DSVL tool.

Activity 1

In order to answer research question 3 (Section 5.3), the first semi-structured interview covered the following initial topics:

1. A short one or two sentence summary of the environmental modeling work the participant is involved in
2. Whether the terms context and provenance were used by or meant anything to the participant in relation to their work in environmental modeling, and if those terms were used then what they meant to the participant
3. A short description of the definition of context and provenance as used in this research, followed by whether these types of information were considered in the participant’s work and if so how these types of information were recorded

Activity 2

The guided walkthrough of the extended DSVL tool involved a brief demonstration of the tool using an example scenario, explaining how the tool can be used to set up, describe, run, and record additional information about the simulation experiment. Participants were encouraged to ask questions and interact with the tool. The scenario used was the same scenario as for the previous study (see Chapter 3 and Appendix F), extracted from ‘Land-use change and subalpine tree dynamics: colonization of Larix decidua in French subalpine grasslands’, Albert et al. [3]. Pertinent information was condensed into a single one page summary with the assistance of one of the original authors.
5.4. METHODOLOGY

Activity 3

In order to address research questions 1 and 2 (Section 5.3), participants were asked to rate the tool. The same measurement scale items from the previous study (Chapter 3) were used. These measurement scale items consisted of those from the TAM[25], in which users rate their perceived usefulness and perceived ease of use for each of the tools. These ratings have been shown to have a strong correlation with the user’s intention to use or acceptance of the subject technology [25]. The measurement scale items were used without modification, with participants being asked to rate each of the statements considering environmental modeling as the ‘job’ referred to in the statements. As with the previous study, the third additional scale for perceived modeling utility was used in order to determine specific views towards understanding and communication. The statements for perceived modeling utility were developed in conjunction with two experienced investigators.

Further details of the measurement scale items used and the rationale for their inclusion can be found in Section 3.4.3.

The results from this activity were used to compare the extended DSVL tool with the two tools described in Section 3.4.4, which were the Original DSVL tool and the Baseline tool. The Extended DSVL tool was compared with the Original DSVL tool in order to answer research questions 1 and 2 (Section 5.3). The comparison with the Baseline tool was performed in order to ensure that the Extended DSVL tool compared favourably with the Baseline tool baseline as established in the previous study.

Activity 4

The final activity, the second semi-structured interview, was used to check, clarify, and expand upon the results of the other activities. This covered the following topics:

1. Clarification and expansion on their questionnaire answers, focusing on those which stood out (were different from other responses within the same group)

2. Overall impressions of the tool

3. Usefulness and utility of including context and provenance information in the same interface as the simulation specification
4. Possible improvements or features which would increase the usefulness, ease of use, or modeling utility

5.4.4 Extended DSVL tool

The DSVL analysed in this study, referred to as the 'Extended DSVL tool', was extended from the DSVL used in the previous study (see Section 3.4.4). The original DSVL tool, and therefore also this Extended DSVL tool, was developed in Eclipse using the GEMS[124] framework. To avoid having to explain the specific concepts of DSVLs and confusing participants, the Extended DSVL tool was referred to as the 'Diagram-based Tool' in all materials presented to them.

Figure 5.2 shows a screenshot of the Extended DSVL tool, depicted with a complete specification of the European Larch scenario used in this study including sample annotations for publication references and parameters of interest.

The DSVL which was presented in the previous study only provided a different way for scientists to interact and specify the environmental simulation. In order to improve the DSVL based on the information gained from the previous study, annotation elements were added to the DSVL. In this prototype, these annotation elements consisted of a textural name, type, and content, although it would be possible to also have annotations which contain URLs, images, datasets, or other information. The annotation elements can be linked to any element within the DSVL, including other annotations.

5.4.5 Statistical analysis approach

For comparison of the measurement scale items from the TAM[25] as well as those for perceived modeling utility, the Wilcoxon rank sum test[125] was chosen for statistical comparison. The Wilcoxon rank sum test is a difference test which assesses whether the population mean ranks of two non-paired sample sets differs, even if the sample sizes of the two sets are different. The Wilcoxon rank sum test was chosen as the data from the Likert scale[81] used for each of the measurement scale items is non-continuous and therefore non-parametric. The R software environment [100] was used to perform the statistical analysis.

A two-tailed test was performed in order to test whether the two types of tool were significantly different in either direction, in order to see if the Extended DSVL tool differed either positively or negatively in comparison to the original DSVL tool and the Baseline tool. The null hypothesis for each comparison was
Figure 5.2: Extended DSVL tool
that there was no difference between the ratings for the two tools. The results were considered significant for p<.05.

In all cases, the difference between ratings was calculated by subtracting the rating for the DSVL or Baseline tool from the rating for the Extended DSVL tool. The median of these differences is reported in Tables 5.1, 5.2, 5.3, 5.4, 5.5, and 5.6.

5.5 Results and discussion

In this section, the results of this study are presented and discussed. First, the qualitative comments collected in the semi-structured interview of Activity 1 (Section 5.4.3), regarding the current practices of environmental scientific modelers in regard to context and provenance information, are covered. Second, the acceptance of the Extended DSVL tool is analysed, using the results of the survey in Activity 3 (Section 5.4.3), along with comparing this data to the results collected in the previous experiment (Chapter 3). Third, the qualitative comments collected in the semi-structured interview of Activity 4 (Section 5.4.3), regarding the usefulness and ease of use of describing and storing context and provenance information within the Extended DSVL tool, are examined. Finally, threats to the validity of this study are discussed.

5.5.1 Qualitative comments - current practices for context and provenance

The first semi-structured interview gathered information about the current practices of environmental scientific modelers in regard to context and provenance information. In particular, firstly whether they considered these types of information to be important, and how they currently went about recording it. Figure 5.3 shows a summary of the comments gathered from the participants, categorised into the following broad categories:

**Considers Context** - context information is thought about and considered to be important by the participant.

**Considers Provenance** - provenance information is thought about and considered to be important by the participant.
5.5. RESULTS AND DISCUSSION

Figure 5.3: Qualitative Comments regarding Context and Provenance

**Regularly Records Context** - context information is regularly recorded in some manner by the participant.

**Regularly Records Provenance** - provenance information is regularly recorded in some manner by the participant.

**Structured Recording** - context and/or provenance information is recorded in a structured format. Examples include formal experiment proposals, provenance databases, or metadata templates attached to datasets.

**Uses Diagrams** - some kind of diagram is used by the participant to record context and/or provenance information. Examples include flowcharts or influence graphs.

**Included in Publications** - context and/or provenance information is included in publications by the participant.

It should be noted that the terms context and provenance which are used here are not widely used by the environmental scientific modelers who participated in this study. Although the concepts of context and provenance are considered by all or most of the participants respectively, in general these types of information are
not given specific titles or labels. Provenance information is sometimes referred to as metadata, but amongst the participants this was not a commonplace term.

All 17 participants thought about context information and considered it important when developing environmental models or simulations. Such context information included knowing the reason for developing the environmental model or simulation, the questions it aimed to answer, the situation in which the environmental model or simulation is applicable, and relevant information which affected the choices made during the development of the environmental model or simulation.

Thirteen participants thought about provenance information, and considered it important. The types of provenance information considered by participants included paper references, dataset sources, and dataset metadata.

Although all 17 participants thought about context information, only seven described regularly recording such information. Most often, context information was not recorded and was simply something which was only thought about and kept inside the scientist’s minds. Of those who recorded context information, these recordings took the form of informal notes, comments in or attached to the software code, work logs, or project proposals.

In contrast to the disparity between consideration and recording of context information, all of the 13 participants who thought about provenance information also recorded it. This relationship between thinking about and recording provenance information intuitively makes sense, as it would be difficult to remember the details about many different papers and datasets at the same time. This difficulty is particularly evident when the provenance of source material does not generally have a direct impact on how it is used. Provenance information was recorded through informal notes, comments attached to the software code or source data files, and more structured methods described below.

Only four participants described recording either context or provenance information in a structured manner. These methods included formal project proposals, archives of the complete project contents including all relevant information, and/or dataset metadata databases or formalised documentation. Of note is that no participant mentioned using any system to record provenance, such as the examples listed in section 5.2. The only software system used to support provenance recording was a database of dataset metadata, which was used by only one participant.

A number of issues were identified as being caused by a lack of context and provenance recording. These issues are summarised below, with the number in
5.5. RESULTS AND DISCUSSION

Parentheses indicating how many participants made the comment.

Participants indicated that a lack of context recording resulted in the following problems:

- Difficulty in accurately writing up experimental reports or publications (2)
- Difficulty in revisiting and understanding past experiments (2)
- Unclear reasons for particular experimental design choices and changes (2)
- Unknown applicability of experiment (1)

Participants indicated that a lack of provenance recording resulted in the following problems:

- Unknown authenticity and reliability of source data (2)
- Unknown authenticity and reliability of third-party code (1)
- Unclear reasons for particular parameter value choices (1)
- Inability to retrieve original source data for further examination (1)

Participants were also asked if they used diagrams, such as flowcharts or other drawings, to describe their experiments or the context and provenance information. Nine (9) participants, or just over 50%, indicated that they did so at least sometimes. These diagrams consisted of flow charts, decision trees, maps, schematics, or influence networks.

Finally, 15 participants indicated that they included context and/or provenance information in their publications, including technical reports. Nine of these participants did not regularly record context, and four did not regularly record provenance. For these participants who did not record context and/or provenance information but did include it in their publications, the necessary information was considered ‘known’ and able to be reproduced from memory and being reminded by revisiting the code and results of the experiment.

Overall, both context and provenance were not generally treated in a formal, systematic fashion when they were considered at all. These results support Hypothesis 5.3.
5.5.2 Acceptance of extended DSVL tool

Perceived usefulness

The distribution of results for each of the measurement scale items for perceived usefulness is shown in Figure 5.4. The median result for all six measurement scale items was 'slightly likely', indicating that the Extended DSVL tool was perceived to be useful overall.

A comparison of the results for perceived usefulness of the Extended DSVL tool and the Original DSVL tool (from Chapter 3) is shown in Table 5.1. Of the six measurement scale items, two showed a significant \( (p<0.05) \) difference, with the Extended DSVL receiving lower ratings for 'Makes Job Easier' and 'Useful'. Three of the remaining measurement scale items also showed that the Extended DSVL received lower ratings than the Original DSVL, but these differences were not large enough to be considered significant. These results indicate that the Extended DSVL tool may be perceived to be less useful than the Original DSVL tool for the purposes of environmental modeling. The implications of this result are further discussed in Section 5.6.2.

A comparison of the results for perceived usefulness of the Extended DSVL
5.5. RESULTS AND DISCUSSION

Table 5.1: Perceived usefulness - difference between Extended DSVL and Original DSVL tools

<table>
<thead>
<tr>
<th>Scale Items</th>
<th>p (two-tailed)</th>
<th>median of differences (Extended-Original)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work More Quickly</td>
<td>0.15</td>
<td>-1.0</td>
</tr>
<tr>
<td>Job Performance</td>
<td>0.10</td>
<td>-1.0</td>
</tr>
<tr>
<td>Increase Productivity</td>
<td>0.052</td>
<td>-1.0</td>
</tr>
<tr>
<td>Effectiveness</td>
<td>0.46</td>
<td>0.0</td>
</tr>
<tr>
<td>Makes Job Easier</td>
<td>0.045</td>
<td>-1.0</td>
</tr>
<tr>
<td>Useful</td>
<td>0.0048</td>
<td>-1.0</td>
</tr>
</tbody>
</table>

Table 5.2: Perceived usefulness - difference between Extended DSVL and Baseline tools

<table>
<thead>
<tr>
<th>Scale Items</th>
<th>p (two-tailed)</th>
<th>median of differences (Extended-Baseline)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work More Quickly</td>
<td>0.96</td>
<td>0.0</td>
</tr>
<tr>
<td>Job Performance</td>
<td>0.29</td>
<td>0.0</td>
</tr>
<tr>
<td>Increase Productivity</td>
<td>0.48</td>
<td>0.0</td>
</tr>
<tr>
<td>Effectiveness</td>
<td>0.58</td>
<td>0.0</td>
</tr>
<tr>
<td>Makes Job Easier</td>
<td>0.32</td>
<td>1.0</td>
</tr>
<tr>
<td>Useful</td>
<td>0.95</td>
<td>0.0</td>
</tr>
</tbody>
</table>

tool and the Baseline tool (from Chapter 3) is shown in Table 5.2. None of the six measurement scale items showed a significant (p<0.05) difference.

Perceived ease of use

The distribution of results for each of the measurement scale items for perceived ease of use are shown in Figure 5.5. All of the measurement scale items received high ratings, with a median rating of ‘quite likely’ for all except for ‘controllable’ (which received a median of ‘slightly likely’).

A comparison of the ratings for perceived ease of use between the Extended DSVL tool and the Original DSVL tool (from Chapter 3) is shown in Table 5.3. None of the measurement scale items showed a statistically significant difference
Figure 5.5: Perceived ease of use of Extended DSVL tool

Table 5.3: Perceived ease of use - difference between Extended DSVL and Original DSVL tools

<table>
<thead>
<tr>
<th>Scale Items</th>
<th>p (two-tailed)</th>
<th>median of differences (Extended-Original)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to Learn</td>
<td>0.24</td>
<td>0.0</td>
</tr>
<tr>
<td>Controllable</td>
<td>0.19</td>
<td>0.0</td>
</tr>
<tr>
<td>Clear &amp; Understandable</td>
<td>0.51</td>
<td>0.0</td>
</tr>
<tr>
<td>Flexible</td>
<td>0.87</td>
<td>0.0</td>
</tr>
<tr>
<td>Easy to Become Skillful</td>
<td>0.96</td>
<td>0.0</td>
</tr>
<tr>
<td>Easy to Use</td>
<td>0.73</td>
<td>0.0</td>
</tr>
</tbody>
</table>

between the two tools.

A comparison of the ratings for perceived ease of use between the Extended DSVL tool and the Baseline tool (from Chapter 3 is shown in Table 5.4. The Extended DSVL tool was perceived to be significantly (p<0.05) better for one measurement scale item, 'easy to use'. The remaining five measurement scale items showed no significant difference between the two tools.
5.5. RESULTS AND DISCUSSION

Table 5.4: Perceived ease of use - difference between Extended DSVL and Baseline tools

<table>
<thead>
<tr>
<th>Scale Items</th>
<th>p (two-tailed)</th>
<th>median of differences (Extended-Baseline)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to Learn</td>
<td>0.28</td>
<td>0.5</td>
</tr>
<tr>
<td>Controllable</td>
<td>0.61</td>
<td>0.0</td>
</tr>
<tr>
<td>Clear &amp; Understandable</td>
<td>0.059</td>
<td>1.0</td>
</tr>
<tr>
<td>Flexible</td>
<td>0.15</td>
<td>1.0</td>
</tr>
<tr>
<td>Easy to Become Skillful</td>
<td>0.978</td>
<td>1.0</td>
</tr>
<tr>
<td>Easy to Use</td>
<td>0.017</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Perceived modeling utility

The six measurement scale items for perceived modeling utility are as follows:

1. I would easily understand the scenario when using the diagram-based tool
2. The diagram-based tool would give me a clear view of the scenario
3. The diagram-based tool would allow me to explore the scenario easily
4. I would be able to grasp the scenario structure easily using the diagram-based tool
5. Using the diagram-based tool would make it easy to communicate the scenario with others
6. I would easily understand a different scenario if it were presented to me using the diagram-based tool

The distribution of results for each of the measurement scale items for perceived modeling utility is shown in Figure 5.6. All of the measurement scale items were rated positively overall with a median score of ‘quite likely’.

A comparison of the ratings for perceived modeling utility between the Extended DSVL tool and the Original DSVL tool (from Chapter 3) is shown in Table 5.5. None of the measurement scale items showed a statistically significant difference between the two different tools.

A comparison of the ratings for perceived modeling utility between the Extended DSVL tool and the Baseline tool (from Chapter 3) is shown in Table 5.6.
Figure 5.6: Perceived modeling utility of Extended DSVL tool

Table 5.5: Perceived modeling utility - difference between Extended DSVL and Original DSVL tools

<table>
<thead>
<tr>
<th>Scale Items</th>
<th>p (two-tailed)</th>
<th>median of differences (Extended-Original)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easily Understand Scenario</td>
<td>0.20</td>
<td>0.0</td>
</tr>
<tr>
<td>Clear View of Scenario</td>
<td>0.60</td>
<td>0.0</td>
</tr>
<tr>
<td>Explore Scenario Easily</td>
<td>0.88</td>
<td>0.0</td>
</tr>
<tr>
<td>Grasp Scenario Structure Easily</td>
<td>0.91</td>
<td>0.0</td>
</tr>
<tr>
<td>Easy to Communicate Scenario</td>
<td>0.91</td>
<td>0.0</td>
</tr>
<tr>
<td>Understand Different Scenario</td>
<td>0.95</td>
<td>0.0</td>
</tr>
</tbody>
</table>

The Extended DSVL tool was perceived to be significantly (p<0.05) better for five of the six measurement scale items.

Overall acceptance

The Extended DSVL tool received overall positive scores for all of the measurement scale items for perceived usefulness and perceived ease of use. As perceived
Table 5.6: Perceived modeling utility - difference between Extended DSVL and Baseline tools

<table>
<thead>
<tr>
<th>Scale Items</th>
<th>p (two-tailed)</th>
<th>median of differences (Extended-Baseline)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easily Understand Scenario</td>
<td>0.090</td>
<td>1.0</td>
</tr>
<tr>
<td>Clear View of Scenario</td>
<td>0.0074</td>
<td>2.0</td>
</tr>
<tr>
<td>Explore Scenario Easily</td>
<td>0.010</td>
<td>2.0</td>
</tr>
<tr>
<td>Grasp Scenario Structure Easily</td>
<td>0.0017</td>
<td>2.0</td>
</tr>
<tr>
<td>Easy to Communicate Scenario</td>
<td>0.00051</td>
<td>2.0</td>
</tr>
<tr>
<td>Understand Different Scenario</td>
<td>0.0023</td>
<td>3.0</td>
</tr>
</tbody>
</table>

Usefulness and perceived ease of use are strongly correlated with intention to use and acceptance [24], these results support Hypothesis 5.1. However, these ratings were not statistically significantly different from the ratings for the Original DSVL tool, and therefore do not show that the Extended DSVL tool is more accepted than the Original DSVL tool and do not support Hypothesis 5.2.

5.5.3 Qualitative comments - context and provenance in the extended DSVL tool

The Extended DSVL tool allows users to record context and provenance in the annotations, which are separate elements on the experimental specification diagram. Fourteen out of the 17 participants gave positive comments regarding the use of these annotations. These comments were as follows, with the number of participants who made the comment shown in parentheses:

- Context information assists the understanding and initial orientation within an experiment (3)
- The annotations allow documentation as the experiment is developed, without changing tools (1)
- Having annotations present in the experimental specification helps remind people to record the context and provenance information (1)
- Annotations force people to consider context and provenance information and think about what they are doing (1)
• Useful to explain changes or sensitivity of parameters (1)
• People are reassured by the inclusion of context information (1)

Participants were also asked to suggest any possible improvements to the tool, with the following comments recorded (likewise, the number in parentheses is the number of participants who made the comment):

• Force annotations to be included (3)
• Include annotations within elements, rather than as separate elements, in order to improve the connection between the information and the experiment (2)
• Add a history of past experiment runs which automatically records the experimental setup, annotations, and results (1)
• Imposing a defined structure on annotations may be useful for improved communication, rather than having a generic annotation which can include any text (1)

No participant responded negatively to the annotations, although three participants did not present an opinion either way.

5.5.4 Threats to validity

The comparison of the Extended DSVL tool and the two tools from Chapter 3 relies on the characteristics of the two sample groups being comparable. Both groups were formed of environmental modellers, all were from respected universities. However, the sample group of this study consisted of a larger proportion of academics with greater modeling experience. Participants for this study were drawn from multiple Australian universities, including the ANU. All of the universities involved are highly reputable and perform world-class environmental science research.

The walkthrough of of the tool was provided by a single researcher, introducing an element of experimenter bias into the study. Care was taken to explain the tools in a dispassionate manner (in the same manner as performed in the study presented in Chapter 3), with instructions restricted to the order of tasks and the location of the relevant interface elements.
5.5.5 Summary of results compared to hypotheses

A summary of the results of this study compared with the hypotheses presented in Section 5.3 are presented below:

Hypothesis 5.1. A DSVL which allows recording of context and provenance information would be accepted by environmental scientific modelers.

Hypothesis 5.1 is supported by the results of this study. All of the measurement scale items for perceived usefulness and perceived ease of use of the Extended DSVL tool received positive results from respondents, indicating that the Extended DSVL tool is accepted by environmental scientific modelers.

Hypothesis 5.2. A DSVL which allows recording of context and provenance information would be more acceptable to environmental scientific modelers than a DSVL which does not.

Hypothesis 5.2 is not supported by this study, as the results for perceived usefulness and perceived ease of use (which are strongly correlated with intention to use and acceptance [24]) are not significantly different between the Extended DSVL tool and the Original DSVL tool.

Hypothesis 5.3. Environmental scientific modelers currently consider and record context and provenance information in an informal and ad-hoc manner, and systematic recording of context and provenance information is not performed.

Hypothesis 5.3 is supported, as the qualitative comments from respondents regarding context and provenance information indicate that if such information was considered at all, it was generally not treated in a formal, systematic manner.

5.6 Implications

The results of this study present implications for two particular areas - first, the current practices of how environmental scientific modelers deal with context and provenance information, and second the acceptance of the Extended DSVL tool for recording such information. Each of these areas will be discussed in turn.

5.6.1 Current practices for context and provenance

The qualitative comments gathered regarding current practices for recording and presenting context and provenance indicate that although the labels of 'context'
and 'provenance' are not used, these types of information are thought about by most or all of the environmental scientific modelers involved in this study. With context information considered by all of the participants, and provenance information considered by 13 out of 17, both types of information are present in the work of an environmental scientist. When the concepts of context and provenance information as used in this thesis were described, all of the environmental scientific modelers were able to understand and - excepting the four participants who did not consider provenance information - provide examples of such information in their own work. The lack of a label or term given to context and provenance information indicates that environmental scientific modelers lack a common and consistent framework in which to consider and treat such information.

Based on these results, there is a mismatch in the consideration and the recording of context information. Although all 17 participants considered context information, only 7 recorded such information with regularity. Context information forms part of the body of scientific knowledge, as it provides the reasons for performing a particular experiment, the parameters considered important, and any background information. The lack of recording of such information presents issues for future use of the work done, as these elements of the body of scientific knowledge will most likely be lost without any clear records being kept. The context information is required to accurately place the results of an experiment within the body of scientific knowledge. According to the system of scientific research presented in Chapter 4, the context information of an experiment within the body of scientific knowledge is required to understand the applications, implications, and limitations of the experiment in regard to future questions, hypotheses, experiments, and results analysis.

In contrast to context information, provenance information was recorded by all 13 of those environmental scientific modelers who thought about this type of information. It could be hypothesised that this is because provenance information is much harder to remember, as given only a dataset it is difficult to guess the source, limitations, or even when the information was produced.

Structured recording of context and/or provenance information was only performed by four participants. For those four participants, there were varied reasons and methods of recording, from self-imposed developer diaries for personal use, to government mandated database records for dataset provenance information. Other participants who did not structured record context and/or provenance information used ad-hoc notes, occasional reports, or no records until a paper is written, when the information is transcribed from memory.
Those participants who stated they used diagrams to record or communicate context and provenance information only used diagrams occasionally. Out of the 13 who recorded context and/or provenance information, nine participants stated they used diagrams at least occasionally (all 9 participants who recorded context information also recorded provenance information), but no participant indicated that the use of diagrams was commonplace. Participants stated that the diagrams which they did draw covered both context and provenance information. For context information, the construction of an environmental experiment was sometimes described using flow charts, decision trees, and influence networks being used. For provenance information, maps were used to record or convey spatial data. This occasional use of diagrams indicates that environmental scientific modelers are willing to use visual representations when they are suitable - situations where the information being conveyed has complex relationships or is location sensitive.

This lack of systematic recording of context and/or provenance information is at odds with the fact that 15 out of 17 participants commonly included context and/or provenance information in their publications. The majority of participants thought about context and provenance information and included one or both in their publications. In comparison, only a minority regularly record context information, and less than 1 in 4 participants used some structured method to recording either type of information. Where context and provenance information was not recorded, its inclusion in publications was generated primarily from memory. The lack of recording sometimes results in repeated experiments when scientists cannot remember thoughts, processes, or the source of data. It also presents accuracy and completeness issues, which impacts negatively on the repeatability of published experiments.

5.6.2 Acceptance of the extended DSVL tool

This section discusses the acceptance of the Extended DSVL tool in comparison to the Original DSVL tool and the Baseline tool. The qualitative comments gathered from participants regarding acceptance of the Extended DSVL tool for recording context and provenance are also discussed.

Comparison with the original DSVL tool

Regarding the quantitative results, of the 18 measurement scale items for perceived usefulness, ease of use, and modeling utility, only two showed a statistically
significant difference (p<0.05) between the Extended DSVL tool and the Original DSVL tool. The two measurement scale items were for perceived usefulness, where the Extended DSVL tool received lower ratings compared to the Original DSVL tool.

The lower ratings for two of the six measurement scale items for perceived usefulness may be explained by the choice of task referred to as the 'job' in the questionnaire. The 'job' was defined as environmental modeling, and the two measurement scale items in question were 'Using [Tool] would make it easier to do my job' and 'I would find [Tool] useful in my job'.

Context and provenance information aids in the understanding, exploration, and communication of an experiment, and the previous study (Chapter 3) demonstrated that environmental scientific modelers consider these activities as different from environmental modeling in general. Therefore, the addition of context and provenance recording to the DSVL adds features which do not directly aid the 'job' of environmental modeling as viewed by environmental scientific modelers. The additional, seemingly superfluous, features included in the Extended DSVL tool may have caused environmental scientific modelers to consider the Extended DSVL tool as less useful for and making it more difficult to perform the 'job' of environmental modeling. However, although the Extended DSVL tool received significantly lower ratings than the Original DSVL tool for those two measurement scale items, the results for perceived usefulness are still overall positive. This indicates that environmental scientific modelers still perceive the Extended DSVL tool as being useful.

It was hypothesised that the Extended DSVL tool, with its added annotations, would be more accepted than the Original DSVL tool which lacked these features. Although the quantitative data does not support this hypothesis that acceptance would be improved, the qualitative comments gathered from the participants indicate that 14 out of the 17 participants indicated they considered the annotations to be useful. From the results of the previous study, it was seen that overall usefulness and modeling utility were perceived differently and independently. The qualitative comments referred to regarding annotations being used as an information recording and communication tool, and thus these comments relate to the perceived modeling utility of the tool. These qualitative comments indicate that there is a difference in the perceived modeling utility of the two tools, as annotations were considered to be useful, but this difference was too small to be detected in this study.

As with the study presented in Chapter 3, particular quirks or other elements
5.6. IMPLICATIONS

of the Extended DSVL tool may have had an impact upon the results, particularly for perceived ease of use. To mitigate this, participants were walked through the usage of the Extended DSVL tool in the same way as for the Original DSVL and Baseline tools, in order to ensure that all three tools were provided equal treatment and participants did not get stuck at any point. The Extended DSVL tool uses the same interface mechanics and interface design as the Original DSVL tool (the Extended DSVL tool adds elements to cover recording context and provenance information, however interaction with those elements is identical to all other elements).

Comparison with the baseline tool

The Extended DSVL tool was not perceived to be different from the Baseline tool for the categories of perceived usefulness or perceived ease of use. None of the six measurement scale items for perceived usefulness showed a significant difference between the Extended DSVL tool and the Baseline tool. One of the six measurement scale items for perceived usefulness, ‘I would find the [Tool] easy to use’, showed a significant difference, where the Extended DSVL tool was rated higher than the Baseline tool.

For perceived modeling utility, the Extended DSVL tool received significantly higher ratings than the Baseline tool for five of the six measurement scale items, with only ‘I would easily understand the scenario when using the [Tool]’ not showing a significantly different result between the two tools. Thus, comparing these results to those of the previous study (Chapter 3), the Extended DSVL tool retains the benefits of the Original DSVL tool by improving perceived modeling utility.

Recording context and provenance in the DSVL

Although there were a number of potential improvements which were suggested by participants, the qualitative comments are overwhelmingly positive regarding the use of the Extended DSVL tool to record context and provenance. The potential of a tool such as the Extended DSVL tool which supports the specification of an experimental design alongside the recording of pertinent context and provenance information was easily identified by the participants. By having such functionality, the act of recording context and provenance information was clearly and explicitly also part of the experiment. This acts as a reminder to scientists to record such information while creating and modifying their exper-
imential design. In addition, it allows them to record the information without having to use an additional tool. This encourages the recording of context and provenance information by scientists.

By encouraging the recording of context and provenance and making such activities more widely performed, the issues (as identified by environmental scientific modelers in Section 5.5.1) caused by a lack of recording such information can be avoided.

## 5.7 Conclusion

In summary, the conclusions which can be drawn from this study are:

1. For the purposes of recording context and provenance information, a DSVL which allows environmental scientific modelers to record such information receives positive comments.

2. For the purposes of environmental modeling, a DSVL which provides the ability to record context and provenance information is accepted by environmental scientific modelers, but is not more or less acceptable compared to a DSVL which does not allow the recording of such information.

3. Currently, environmental scientific modelers rarely record context and provenance information in a formal or systematic manner. Apart from databases of dataset metadata, there was no use of any software systems to support the recording of context or provenance information.

The result that environmental scientific modelers do not find a DSVL which allows the recording of context and provenance information any more or less acceptable than a DSVL which does not may be explained by the last conclusion, where environmental scientific modelers rarely record context and provenance information in a formal or systematic manner. This current lack of formal and systematic recording could mean that there is no perceptible difference between a tool which lets them record such information and a tool which does not - since they do not currently record such information then there may be no perceived need to have such features in a tool. Thus, the presence or absence of such features does not affect the perceived usefulness or ease of use of the tool, as those features may not be used at all in the first place.

However, environmental scientific modelers recognise that a lack of recorded context and provenance information causes a number of problems. A lack of
recorded context information causes difficulties in accurately producing publications, revisiting past experiments, and understanding experimental designs and applicability. A lack of recorded provenance information results in unknown reliability of source data and third-party code, unclear reasons for parameter values, and inability to retrieve and examine the source data used. Environmental scientific modelers acknowledge that improving the manner in which context and provenance is recorded and making such activities more commonplace would reduce such problems.

The Extended DSVL tool received positive ratings overall for perceived usefulness and perceived ease of use, indicating that environmental scientific modelers would be willing to use it. However, the Extended DSVL tool did not demonstrate increased acceptability in comparison to the Original DSVL tool nor the Baseline tool. The overall usefulness of the Extended DSVL tool is supported by the collected qualitative comments, with no negative comments being recorded regarding context and provenance recording using the Extended DSVL tool.
CHAPTER 5. DSVLS, SCIENTISTS, CONTEXT, AND PROVENANCE
Chapter 6

Conclusion

6.1 Summary

The initial motivation of this research was to determine how software engineering could be of aid to scientists. In order to address this motivation and the aim of improving the effectiveness of scientific software development (Section 1.4), the following activities were performed: an investigation of current software engineering practices in scientific software development, the acceptance of DSVLs by environmental scientists, and the tailoring of DSVLs for use by environmental scientists.

This chapter first summarises the contribution made by this thesis. Second, the limitations of the research presented in this thesis are described. Third, a summary of the implications of this research is presented. Fourth, avenues for potential future work are explored. Finally, the overall conclusions of this thesis are presented.

6.2 Contribution

The main contributions of the research presented in this thesis are as follows:

- The measurement and analysis of the use of software engineering tools, techniques, and methodologies in scientific software development, and the discovery that the use of said tools, techniques, and methodologies in scientific software development has improved. However, there remains room for improvement.
• The identification of the relative importance of various non-functional requirements by scientific software developers.

• The demonstration that a DSVL tool has clear, perceptible benefits in modeling utility for environmental scientists.

• The discovery that the established benefits of DSVLs are not evident to and/or considered useful by environmental scientists.

• The discovery that there exists a gap between the consideration and the treatment of context and provenance information by environmental scientists.

• A definition of the system of scientific research

• The discovery that environmental scientists respond positively to a tool that integrates the recording of context and provenance information alongside an experimental setup and specification.

• The demonstration that a software engineering methodology can be applied and would be accepted in a non-software specific area, namely the use of a DSVL to record context and provenance information.

6.3 Limitations

Although the limitations of each individual study have been explored in the relevant chapters, there are some limitations of the research presented in this thesis as a whole. The limitations which apply to this research as a whole and in particular to the contributions of this research are as follows:

• The two studies involving DSVLs (Chapters 3 and 5) only test one particular non-DSVL tool and one particular DSVL tool (of which a tailored version is also tested). This limits the ability to generalise the results of this research. However, both tools are broadly representative of tools of their type, which means that generalisation is possible to a certain degree. Further research would be able to confirm this generalisation.

• The participant pools for both DSVL studies (Chapters 3 and 5) were comprised solely of environmental scientists. This limits the applicability of the conclusions and contributions of this research to the environmental sciences field.
6.4 Implications

The implications of individual parts of this research have been presented throughout this thesis. There are a number of main implications which will be summarised in turn.

6.4.1 Problems with scientists' acceptance of DSVLs

The survey presented in Chapter 2 measured and analysed the use of software engineering tools, techniques, and methodologies in scientific software development, and found that such use was more widespread than previously reported. However, gaps still remain, even though the use of such software engineering tools, techniques, and methodologies would improve software quality, particularly in the areas considered important by scientific software developers. These gaps include a lack of consistent tool use, documentation, and testing and verification activities.

The study presented in Chapter 3, which compares environmental scientists' acceptance of a new DSVL tool and an existing Form tool for specifying environmental simulations, demonstrates that there are issues in marketing the use of DSVLs to environmental scientists. These issues arise because of a mismatch between what scientists perceive to be the benefits of a DSVL tool, and what scientists require or desire from software systems in general. As identified and discussed in Chapter 4, software engineering has thus far focused on how software works and how it is built, whereas scientists are more interested in the role of software within the broader system of scientific research. The issues identified are:

- Lack of perceived usefulness of the DSVL
- Lack of perceived need for the DSVL

- No analysis or measurement of the actual use has been performed for the tools described and tested in this thesis. Acceptance, or intention to use, is highly correlated with actual use [26], however they are not equivalent. Therefore, although the results reported in this thesis are indicative of actual use of the tools should they be introduced, this should be confirmed with further study.
- Lack of usability of the DSVL
- Difficulty integrating the DSVL into existing workflows

Each of these issues will be examined in turn.

**Lack of perceived usefulness of the DSVL**

The first issue, the lack of perceived usefulness of the DSVL, arises when the idea being introduced is not seen to provide any benefits. Where the benefits of a idea are unclear or non-obvious, potential users have difficulty finding the value in adopting the idea. DSVLs have been shown to improve reliability, maintainability, productivity, and reusability. However, these benefits are only clearly applicable and have been demonstrated within the development and use of software, and are not seen as immediately worthwhile benefits to the system of scientific research as a whole. Without the value being apparent to potential users, there is little reason for them to adopt the idea. It may be possible to improve the perceived usefulness by having actual real-world implementations of DSVLs, and comparing and contrasting these real-world implementations with existing practices. This could demonstrate that, given the current role of software in scientific research, software engineering ideas also benefit scientific software as a whole.

**Lack of perceived need for the DSVL**

The second issue is a lack of perceived need for the DSVL being introduced. This is where the benefits of the idea are not considered to be useful by the potential user. The DSVL tool received higher ratings for perceived modeling utility compared to the baseline Form tool, indicating that there are some benefits to adopting the DSVL tool which are evident to environmental scientists. However, the acceptability of the DSVL tool in regard to environmental modeling in general was not significantly different to that of the baseline Form tool, with the ratings for perceived usefulness and perceived ease of use not significantly different between the two tools. According to these results, the DSVL tool is seen to be better for understanding, exploring, and communicating the experiment being specified, yet these improvements are not reflected in the tool's acceptability for environmental modeling by environmental scientists. Thus, the DSVL does provide a benefit, but this benefit is not perceived as being needed or being useful.

This lack of perceived need may be due to scientists' focus on scientific research rather than the tools used to produce the research. From the point of
6.4. IMPLICATIONS

view of a scientist, the specific tool that is used for a task may not matter - what matters is the role which that tool performs within the system of scientific research. Provided that the tool provides the required functionality and performs it correctly, the tool is suitable for the job. This can be seen by the DSVL tool, as for the purposes of setting up and specifying an experimental setup it provides the same functionality - although with a different interface - as the Baseline tool. Therefore, for the purposes of scientific modeling, both tools perform the same role and both tools are equally acceptable to scientists.

Lack of usability of the DSVL

The third issue is a lack of usability of the DSVL. A system which has high usability is easy to learn, efficient to use, easy to remember, has few errors, and is subjectively pleasing[94]. If the difficulty of adopting the system outweighs the perceived benefits, then potential users will be - rightfully so - reluctant to adopt said system. Although the TAM [25] does not directly measure usability, all of the elements of usability, with the exception of efficiency, are reflected in the measurement scale items for perceived ease of use. There was no significant difference in the perceived ease of use of the DSVL tool and the Baseline tool. The qualitative comments indicated that both tools have particular advantages and disadvantages in terms of usability, indicating that a system with high usability would take particular elements from both the Baseline and DSVL tools.

Difficulty integrating the DSVL into existing workflows

The final issue is the ability to integrate DSVLs with existing workflows. Potential users will already have ways in which they perform their activities, for example, scientific research for environmental modellers. The DSVL replaces or augments existing processes, but the capabilities of the DSVL may not match what the potential users expect it to be able to do. Even if the capabilities and expectations match, integrating the DSVL into the old processes may be hampered by a mismatch between old and components of the workflow - for example, the DSVL may replace only a part of an existing system, resulting in the need to use both the new and old systems to retain the required functionality. As an example of this issue, when using the DSVL tool, inputting the parameters for each element in the scenario being modeled was seen as clunky and time-consuming in comparison to the Form tool. Therefore, both tools would need to be used in order to firstly describe the scenario (using the DSVL tool) and secondly specify
parameters (using the Form tool) in the easiest manner possible for each activity. However, this use of two different tools for different parts of the workflow requires users to know how to use both tools, as well as how to transfer data between them. The cost and effort involved in implementing this system is much less likely to be considered economical when compared against simply retaining the Form tool for both activities, even if its use is less than ideal for describing the scenario.

6.4.2 Thinking about software engineering within systems

This thesis has shown that software engineering can play a much larger role in the system of scientific research. This idea also applies to other systems as well. As explored in Chapter 4, software engineering has generally focussed on the ‘how it works’ aspects of software - how software is built, how software can be built more efficiently, and how the software itself can be made more efficient. When a systems thinking approach is applied to the systems in which software operates, new roles which software engineering tools, techniques, and methodologies can play within the target system can be identified.

Although software engineering has proven benefits in software development productivity and software quality, these benefits have not been realised in many fields in which software development is a part. As seen in scientific software development and explored in Chapter 2, the acceptance and use of software engineering tools, techniques, and methodologies has been slow, although progress has been made in recent years. This lack of uptake may be caused by a mismatch in focus. As explored in Section 4.2.4, software engineering has traditionally focussed on the development of software as an isolated product. The application of software engineering to scientific software development is justified as valuable to scientists through the flow-on effects of improved confidence and greater volume of scientific results [10, 57, 127]. However, as shown in the study in Chapter 3, scientists are more interested in developing and understanding an experiment and its purpose, rather than the exact details of development and implementation.

In order to address this issue, taking a systems thinking approach to the role of software engineering would be more appropriate. According to the systems thinking approach, the current focus of software engineering, on optimisation of software development and performance, is not ideal. Software engineering tends to focus on improving software and software development, separate from the systems (in which software operates. The systems thinking approach indicates
6.4. IMPLICATIONS

that this naive optimisation of a single component may have minimal or even
detrimental effects on the whole system (Section 4.2.1). Instead, a holistic view
of the role of software within the entire system should be taken.

Using the systems thinking approach, the focus is on the purpose and role of
software and software engineering, providing a means to ensure that the software
and software engineering activities are applied in the most useful, effective, and
accepted ways within the target system.

Expanding the ways in which software engineering ideas can be used provides
many more opportunities for software engineering ideas to improve the systems
in which software is used. This has been shown in the study presented in Chapter
5, where it was shown that environmental scientists accepted a DSVL tool which
allowed them to capture and record context and provenance information alongside
the experimental design.

6.4.3 Environmental scientists recording context and provenance using DSVLs

The study presented in Chapter 5 investigated one particular tailoring of a DSVL-
based tool for environmental scientists, with the addition of context and prove-
nance recording within the tool alongside the existing experiment specification
functionality. This addition was intended to address the first two issues presented
in Section 6.4.1, those being the lack of usefulness and/or the lack of perceived
need for the idea. This broadening of focus, from how the idea affects the de-
velopment of scientific software in particular, to a wider role within the system
of scientific research in general, aimed to improve the acceptance of the idea by
environmental scientists.

The hypothesis of the study was that the ability to record context and prove-
nance information using the same tool as that used to specify an experimental
simulation would be useful for scientists. This hypothesis was based on results
of the previous study in Chapter 3, where it was determined that scientists were
more interested in the context and provenance information involved in the exper-
imental simulation - the parameters of interest, input data, and expected results
- than the specifics of how the experimental simulation was specified.

The effectiveness and potential value of including this type of non-programming
information in the DSVL is shown in the feedback collected from environmental
scientists (see Section 5.5.3). This tailoring did not appear to significantly affect
the perceived usability or ease of use of the DSVL, which indicates that although
it was not a positive influence in regard to these factors the addition of extra features was not a negative influence either. The comments from environmental scientists revealed that the integrated nature of the context and provenance recording features would be considered beneficial.

6.4.4 A lack of context recording by scientists

As was demonstrated by the study presented in Chapter 5, although context information is considered by the vast majority of scientists, the recording of this type of information is not commonly performed. When it is recorded, this recording is rarely performed in a formal or systematic manner. Without any record of the context information, the only repository of the information is inside the scientists' minds, leading to incomplete or inaccurate reporting of such information in publications.

In addition to the issue of imperfect recall of context information, communicating and ensuring a common understanding is also an issue without records of such information. Without an accurate and complete record of context information, each user or viewer of an experiment is left to make their own judgements on the purpose and reasoning behind the work. This can result in each individual having differing views of the goals and purpose behind the scientific experiment being performed.

Both of these issues present a strong case for why recording of context information is important. However, as shown in Chapter 5, little research has been performed in this area. The study demonstrates that although scientists understand the importance of context information, it is not commonly recorded. It can be postulated that the recording of this information is not considered important as this activity - like producing software in general as identified in Chapter 2 - is not integral to the act of scientific research and producing scientific results. Whether the context is recorded or not does not immediately affect the results generated - only when the results are published or used in the future is context important for complete understanding. In a similar way to improving the manner in which scientific software is developed, the improvement of context recording is likely to encounter the same obstacles - the perceived lack of usefulness, the amount of time involved, and a resistance to changing from previously tried and proven methods to new and unproven ones.
6.4.5 A lack of systematic provenance recording by scientists

Provenance information was much more widely recorded by environmental scientists, as shown in the study presented in Chapter 5. Although all of the scientists who considered provenance information also recorded it, it was rarely recorded in a formal or systematic manner. This lack of formal or systematic manner may present issues with the quality of the information collected - without a formal or systematic methodology, the provenance information may be incomplete or inadequate for its required purposes.

Although there have been multiple systems developed to support the recording of provenance information in scientific fields (see Chapter 5), none of the environmental scientists who participated in the study indicated they used any of these or other similar provenance management systems. Although the reasons for this lack of use were not specifically addressed in the previous studies, it can be postulated that recording of provenance information is not considered to be of primary importance to scientists. The time and effort taken to record provenance information while developing and performing an experiment may not be considered immediately valuable by environmental scientists. As identified previously, the primary goal of scientists is to perform scientific research, and recording provenance information does not immediately improve the productivity of scientists.

Provenance information is important for ensuring the reproducibility of scientific experiments[4, 23]. Reproducibility remains an issue in computational science [39]. The benefits of provenance information are mainly later on in the scientific process, after the initial research has been performed and when other scientists seek to reproduce the results. Therefore, recording provenance information formally and/or systematically is perceived by scientists to require too much effort for the gain provided - the informal, ad-hoc recording of provenance information is sufficient for the scientists’ immediate requirements. The lack of focus on provenance recording has been asserted previously [18, 20], and the results presented in this thesis confirm these assertions.

6.5 Future work

The research presented in this thesis poses a number of further questions which could be addressed with further work. The work suggested in this section ad-
dresses the limitations presented in section 6.3, along with possible avenues for the expansion of this research.

6.5.1 Context and provenance recording

The study presented in Chapter 5 contains an initial exploration of the current treatment of context and provenance information by environmental scientists, with the results that context and provenance is thought about, but is not recorded and the impacts of a lack of recording are understood. Further research is required to confirm that these results are more widely applicable. In addition, a more in-depth study into the impacts and implications of a lack of context and provenance recording could also be performed.

6.5.2 Acceptability

The studies presented in Chapters 3 and 5 could be expanded in four ways, which are described below.

1. The acceptability of tools, both DSVL and non-DSVL based, for the purposes of scientific modeling, could be investigated. This would enable a wider comparison of the acceptability of various different modeling tools and generalisation of the results shown in this thesis.

2. The acceptability of the tools in scientific areas other than environmental modeling could also be investigated. This would allow the generalisation of the results to other scientific areas.

3. The Extended DSVL tool (presented in Chapter 5) which can be used to record context and provenance information received positive comments from environmental scientists. However, the tool was not viewed as being more acceptable than the Original DSVL tool (presented in Chapter 3). Further investigation and a cost/benefit analysis of the tools would provide further insights into the reasons behind the levels of acceptance of various tools by scientists.

4. A number of additional features and refinements to the Extended DSVL tool were suggested by scientists. Further development and research is required to apply and test these refinements, and determine their effect on the acceptance of the tool.
6.5.3 Developing DSVLs from domain expert diagrams

The potential use of DSVLs for purposes other than purely software development can be generalised into a new approach for developing DSVLs. Although there has been literature regarding techniques for building DSVLs [67, 79, 86] as well as numerous environments to facilitate their construction [49, 69, 129], the process of developing the syntax and semantics of a DSVL has received limited attention. Repenning and Sumner [101] describe an iterative design process between end users (domain experts) and designers (software engineers), refining and improving the functionality of elements of the DSVL. However, this process begins with a generic visual programming syntax with no domain orientation or elements and the development is driven by how the end users attempt to use this generic visual language. This approach also involves feedback from end users regarding the DSVL currently being developed.

The approach used to develop and tailor DSVLs for environmental scientists in this research differs from the above. The approach used here can be summarised as follows:

1. A DSVL is developed by a software developer, using domain knowledge gained from discussions and working in conjunction with an environmental scientist.

2. Environmental scientists, who are potential users of the DSVL, are asked to produce a diagram which explains the environmental scenario to be modeled.

3. These diagrams are analysed to determine commonalities, and in particular those which differ from the DSVL.

4. The DSVL is modified to accommodate elements which were drawn by the environmental scientists.

The difference between the approach used in this research and that of Repenning and Sumner is that the diagrams produced by potential users are not created using any pre-specified language, visual or otherwise. These diagrams are also created before the potential users are exposed to the DSVL. Instead, these diagrams are what the potential users would normally draw to explain and describe the relevant elements of the domain. The second step of the above process, producing diagrams, is independent of the first step of developing a DSVL as normal.
Thus this first step is not required, instead developing a DSVL directly from the diagrams created by the potential users. The modified process is as follows:

1. Future users produce diagrams which explain the relevant elements of the domain.

2. These drawings are analysed to determine commonalities and elements of interest to the users.

3. A DSVL is created based on those commonalities and elements of interest.

This approach would allow DSVLs to more closely mimic the diagrams already produced by users. In cases where diagrams are not usually produced, this approach gathers information from future users without biasing them towards any particular interface or stylistic choices.

Another advantage of this approach is that it allows information to be gathered about what potential users consider important and might be included in the resulting DSVL. This may include information not directly related to producing software, for example context information for environmental scientists (Chapter 3).

### 6.5.4 Scientific software developers and DSVLs

The research presented in this thesis shows that scientific software users are willing to accept DSVLs. This was investigated on the basis that if users are unwilling to use a software engineering idea, implementing it would be worthless (as it would not be used) regardless of its other benefits. However, in order for DSVLs to exist for scientific software users, scientific software developers must first develop the appropriate DSVLs. There has been some previous development of DSLs in scientific areas (Section 3.2.3). However, investigating and improving the acceptability of such approaches by scientific software developers would allow the usage and uptake of such approaches to be increased. This would have beneficial effects in the quality of software, based on the benefits of DSLs and DSVLs as identified in existing studies (Section 3.2.1).

### 6.5.5 Other ways in which software engineering can help

Further research is possible regarding the use of software engineering tools, techniques, and methodologies to support activities other than software development,
as explored in Chapter 4. This use of tools, techniques, and methodologies could ease their introduction into fields which use and develop software but are not focused on the construction of software, such as scientific research. The use of DSVLs to record context and provenance in Chapter 5 provides one example of a non-software development application of a software engineering tool.

6.6 Overall conclusion

The overall conclusion that can be drawn from the research presented in this thesis is that software engineering can and should play a much greater role within the system of scientific research.

One particular example of such a role is the use of DSVLs to improve the manner in which context and provenance information is considered and recorded. This treatment of context and provenance information inside a DSVL framework differs from existing approaches in that it integrates the recording of context and provenance information alongside experimental design and setup. This encourages experiment developers to explicitly consider and capture context and provenance information, and allows them to directly link such information with components of the experimental design.

The research presented in this thesis provides a basis for future work regarding the construction and use of DSVLs in the area of environmental sciences, which can be expanded for applications in other fields as well. In addition, this research also provides a basis for future work regarding the use of software engineering tools, techniques, and methodologies to support activities other than software development.
CHAPTER 6. CONCLUSION
Appendix A

Survey: Ethics Approval
HUMAN RESEARCH ETHICS COMMITTEE
Application Form

Created by: u4113344
Record number: 2814
Protocol type: Expedited Ethical Review (E1)
Protocol number: 2008/518

Date entered: 28/10/2008
Ethics program type: Postgraduate
Requested start date: 01/01/2009
Requested end date: 31/12/2010

Protocol title: Survey of practices in scientific software development

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flint, Shayne</td>
<td>Supervisor</td>
<td>Research School of Computer Science, College of Engineering and Computer Science, ANU</td>
</tr>
<tr>
<td>Nguyen-Hoan, Luke</td>
<td>Primary investigator</td>
<td>Research School of Computer Science, College of Engineering and Computer Science, ANU</td>
</tr>
<tr>
<td>Sankaranarayana, Ramesh S</td>
<td>Supervisor</td>
<td>Research School of Computer Science, College of Engineering and Computer Science, ANU</td>
</tr>
</tbody>
</table>

Investigators Detailed

Name: Flint, Shayne    Role: Supervisor

Expertise: Shayne Flint is a Senior Lecturer in the Department of Computer Science within the College of Engineering and Computer Science at the Australian National University. Shayne has broad experience in the development and application of software and system engineering methodologies in industry, government and academia. His current work is building upon this background to develop approaches software-intensive systems engineering within multi-disciplinary environments. Shayne is a Chartered Professional Engineer (Member, Engineers Australia) and a member of IEEE.

Name: Nguyen-Hoan, Luke   Role: Primary investigator

Expertise: Luke Nguyen-Hoan is a PhD candidate in the Department of Computer Science, commenced in February 2008. His research area is in software-intensive systems engineering, and is focused on finding ways of improving the development of software for scientific purposes, potentially by the application of existing knowledge in the software engineering field. Luke is currently collaborating with ecological scientists from the Fenner School of Environment and Society at the ANU and the Department of Biology, Ecole Normale Superieure in France on a
HUMAN RESEARCH ETHICS COMMITTEE
Application Form

project to develop an ecological simulations software platform.

Name: Sankaranarayana, Ramesh S  Role: Supervisor

Expertise: Ramesh Sankaranarayan is a Senior Lecturer in the Department of Computer Science within the College of Engineering and Computer Science at the Australian National University. Ramesh's current research interests are in the area of Software-Intensive Systems Engineering and Information Retrieval. He has supervised two PhD students, namely, Mr. Tim Jones (current) and Dr. Tin Tang (completed), who required ethics approval for carrying out experiments that involved the use of human judges to evaluate the relevance and quality of search results returned by search engines.

External Investigators

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
<th>Institution</th>
</tr>
</thead>
</table>

Departments

<table>
<thead>
<tr>
<th>Primary</th>
<th>Department</th>
<th>Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Research School of Computer</td>
<td>College of Engineering and Computer Science</td>
</tr>
<tr>
<td></td>
<td>Science</td>
<td></td>
</tr>
</tbody>
</table>

Project Questions Detailed

Description of Project

Describe the research project in terms easily understood by a lay reader, using simple and non-technical language. This research project is intended to provide insights into the current development practices used by scientific software developers. The research will investigate the scientific and software development backgrounds of the developer and attributes of software previously produced by the developer such as its size, source code language, and user base.

The initial survey is intended to obtain an overall picture of some examples of the current state of scientific software development, and is not intended to provide rigorous statistical results.

Depending upon results of the first survey, further surveys may be conducted. These surveys are anticipated to be conducted in the same manner as the initial survey covered in this ethics application. If subsequent surveys are to be conducted in a differing manner to the initial survey, then new ethics applications will be filed.

Location of Data Collection
APPENDIX A. SURVEY: ETHICS APPROVAL

HUMAN RESEARCH ETHICS COMMITTEE
Application Form

Australia Yes
Overseas No

Provide country / area where data collection will be conducted Data collection will be conducted from the ANU, ACT Australia.

Aims of the Project

List the hypothesis and objectives of your research project. The main objective of this research is to obtain an overall picture of some examples of the current state of scientific software development, and in particular the current use of software engineering development practices in this area.

This research will: (1) determine the effect of current software engineering development practices in use in the scientific software development field; (2) examine why those software engineering development practices which are in use by the scientific software development community have been adopted; (3) identify the areas in which scientific software development may be streamlined through the further introduction of software engineering development practices. The aim of this research is not to obtain a statistically valid result, as all of these objectives will be constrained by the limited sample of scientific software developers surveyed.

The planned survey will seek to answer the three questions posed above, however further investigation may be required in order to expand upon any areas of interest which are identified from the initial survey.

Methodology

In language appropriate for a lay reader, explain why the methodological approach minimises the risk to participants. (For surveys, include justification of the sample size). The risk to participants is minimised through voluntary participation and full disclosure of the purpose and conduct of the survey through the information sheet.

The survey asks participants to answer questions regarding already complete projects, and therefore will not affect any projects which are currently in progress directly.

The sample chosen for the survey is a convenient sample as this survey is intended to get an overall picture and some insights, rather than providing a rigorous statistical sample and results.

To minimise the risk of releasing confidential, sensitive, or information that the participants may not wish to be released, for any publication which uses specific comments from surveys, specific consent regarding the accuracy and use of those comments will be sought from the participant. In addition, if results that may result in the identification of participants are to be published, specific consent regarding the use of the results and possible impact will be sought from the participant.

Provide the survey method, a list of the questions to be asked or an indicative sample of questions. These should give a good sense of the most intrusive/sensitive areas of questioning. Indicative sample of survey questions.

1. How many years of programming experience do you have?
   This question is being asked in order to determine the relative expertise of the survey respondents.

2. In reference to scientific software which you have used or developed, rate the importance of the following nine software quality attributes (a-i) using the following 1 to 5 scale.
HUMAN RESEARCH ETHICS COMMITTEE
Application Form

(1) very important - the software must do this in order to be useful
(2) important - the software should do this in order to be useful, but is still usable without it
(3) neither important nor unimportant - the software may do this but will still be useful without it
(4) unimportant - the software does not actively seek not sacrifice this attribute
(5) very unimportant - the software sacrifices this in order to concentrate on other attributes

a) functionality (the ability to perform all required functions)
b) reliability (the ability to perform with correct, consistent results)
c) maintainability (the ability to be easily corrected)
d) availability (the ability to be accessed and operated when needed)
e) flexibility (the ability to be easily adapted to changing requirements)
f) portability (the ability to be easily modified for a new environment)
g) reusability (the ability to be used in multiple applications)
h) testability (the ability to be easily and thoroughly tested)
i) usability (the ability to be easily learned and used)

3. Are there any attributes which you feel are important for scientific software which are not adequately covered by the above options?

These questions are being asked in order to determine which attributes that should be considered when specifying a software system's quality, as described in a widely used industry standard, are considered most important for scientific software developers, and to determine the appropriateness of this list of attributes to the scientific software development field.

The following two questions will be asked in reference to the three most recent scientific software applications that the respondent has worked on (or less than three if appropriate):

4. What programming language(s) was the program written in?

5. Describe briefly (less than 25 words) why those programming language(s) were chosen

These questions are being asked in order to determine what sorts of programming languages are chosen and what, in the mind of the respondent, drives the choice of those programming languages.

What mechanisms do the researchers intend to implement to monitor the conduct and progress of the research project? For example:

How often will the researcher be in touch with the supervisor?

Is data collection going as expected? If not, what will the researcher do?

Is the recruitment process effective?

How will the researcher monitor participants willingness to continue participation in the research project, particularly when the research is ongoing? The researcher will formally meet with the supervisor weekly, and the supervisor is easily accessible in person if required between the regular weekly meetings.

If the response rate to the survey is low, unexpected answers to questions are received, or data collection is not going as expected in any other way, then the survey content and/or conduct will be reviewed and a variation will be filed as appropriate.

As participation in the survey is voluntary, willingness to participate is demonstrated by return of the survey.

Participants

Provide details in relation to the potential participant pool, including:

target participant group;
identification of potential participants;
initial contact method, and
recruitment method. The target participant group for the survey is anyone who has recently (<5 years) been
directly involved with the development of one or more scientific software applications.

The initial group of potential participants has been identified through work with the Fenner School of Environment and Society at the ANU. Further potential participants will be identified through contact with these and other ecological simulation modellers with which the investigator is also collaborating with.

Initial contact method with the participants will be by email. This initial email will include why they were contacted (who referred them) as well as the information pack.

Proposed number of participants 20

Provide details as to why these participants have been chosen? These participants have been chosen as they are a convenient sample, which is appropriate for the goals of the experiment (to provide overall picture and some insights, rather than providing a rigorous statistical sample and results).

Cultural and Social Considerations/Sensitivities

What cultural and/or social considerations/sensitivities are relevant to the participants in this research project? Cultural considerations are minimal as the research project is a non-interactive survey, the information sheet is provided prior to the survey being administered, and potential participants are given the opportunity to ask questions prior to participation.

Incentives

Will participants be paid or any incentives offered? If so, provide justification and details. Participants will not be paid and no incentives will be offered.

Benefits

What are the anticipated benefits of the research? The anticipated benefits of the research are for the investigator to gain increased understanding of the scientific software development area, the current practices used, and where there is room for improvement of these practices.

To whom will the benefits flow? Although the immediate benefits of this research (that of increased understanding) are for the investigator, this research will be used to identify areas where the practices used in the scientific software development field can be improved. These improvements of more effective and efficient software development practices in the scientific software development field will benefit people involved in the development of scientific software, who are the target group of the survey.

Informed Consent

Indicate how informed consent will be obtained from participants. At least one of the following boxes MUST be ticked ‘Yes’.

In writing No

Return of survey or questionnaire Yes

Orally No

Other No
HUMAN RESEARCH ETHICS COMMITTEE
Application Form

If Oral Consent or Other, provide details.

Confidentiality

Describe the procedures that will be adopted to ensure confidentiality during the collection phase and in the publication of results. Results of the survey and the identities of participants will only be accessible in full to the investigators and supervisors named in this application, excepting cases where the law requires otherwise.

If results of the research are to be reported to participants by email as per the feedback section in this application, participants will be emailed using the bcc field to ensure that email addresses remain private and are not known to other participants.

If specific comments from surveys are to be published, specific consent regarding the accuracy and use of those comments will be sought from the participant.

If results that may result in the identification of participants are to be published, specific consent regarding the use of the results and possible impact will be sought from the participant.

Data Storage Procedures

Provide an overview of the data storage procedures for the research. Include security measures and duration of storage. The data collected from the survey will be stored according to the Department of Computer Science’s storage procedures. Electronically submitted surveys and scanned copies of paper surveys will be burnt to DVD and stored by the Department of Computer Science administration for five (5) years.

Feedback

Provide details of how the results of the research will be reported / disseminated, including the appropriate provision of results to participants. If appropriate, provide details of any planned debriefing of participants. If the results of the survey are used in any publication, then a summary of the information from the survey which was used in the publication and a copy of the publication will be provided by email to all interested participants.

Supporting Documentation

Please ensure electronic copies of any supporting documentation have been uploaded the documents tab of the relevant protocol.

Has this work been approved by another Human Research Ethics Committee (HREC)? No

If yes, please give the name of the approving HREC.
APPENDIX A. SURVEY: ETHICS APPROVAL

HUMAN RESEARCH ETHICS COMMITTEE
Application Form

<table>
<thead>
<tr>
<th>High Risk One Summary</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is this a clinical trial?</td>
<td>No</td>
</tr>
<tr>
<td>Does this research involve the intentional recruitment or issues involving Aboriginal and / or Torres Strait Islander Peoples?</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>High Risk Two Summary</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does this research involve Human Genetics?</td>
<td>No</td>
</tr>
<tr>
<td>Does this research involve Human Stem Cells?</td>
<td>No</td>
</tr>
<tr>
<td>Does this research involve Women who are pregnant and the Human Foetus?</td>
<td>No</td>
</tr>
<tr>
<td>Does the research involve people highly dependent on medical care who may be unable to give consent?</td>
<td>No</td>
</tr>
<tr>
<td>Does the research involve people with a cognitive impairment, an intellectual disability or a mental illness?</td>
<td>No</td>
</tr>
<tr>
<td>Does this research involve an intention to study or expose or is likely to discover illegal activity?</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expedited Questions Summary</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Third Party Identification</td>
<td>No</td>
</tr>
<tr>
<td>Children or Young People</td>
<td>No</td>
</tr>
<tr>
<td>Dependent or Unequal Relation</td>
<td>No</td>
</tr>
<tr>
<td>Membership of a Group, or Related Issues</td>
<td>No</td>
</tr>
<tr>
<td>Physical Harm</td>
<td>No</td>
</tr>
<tr>
<td>Psychological Harm (includes Devaluation of Personal Worth)</td>
<td>No</td>
</tr>
<tr>
<td>Social Harm</td>
<td>No</td>
</tr>
<tr>
<td>Economic Harm</td>
<td>No</td>
</tr>
<tr>
<td>Legal Harm</td>
<td>No</td>
</tr>
<tr>
<td>Covert Observation</td>
<td>No</td>
</tr>
<tr>
<td>Deception</td>
<td>No</td>
</tr>
<tr>
<td>Sensitive Personal Information</td>
<td>No</td>
</tr>
<tr>
<td>Overseas Research</td>
<td>No</td>
</tr>
<tr>
<td>Collection, use or disclosure of personal information WITHOUT the consent of the participant</td>
<td>No</td>
</tr>
</tbody>
</table>
Appendix B

Survey: Scientific Software Development
Section 1. Consent Form (Questions marked * are mandatory)


1. I have read the information sheet for this project and understand its contents. I have had the nature and purpose of the research project, so far as it affects me, fully explained to my satisfaction by the research worker. My consent is freely given.

2. I understand that if I agree to participate in the research project I will be asked to complete a survey. The survey will take approximately 20 minutes to complete and will involve questions about my experiences in the development of scientific software applications.

3. I understand that I may be contacted in the future regarding participation in follow-up surveys. Participation in follow-up surveys is not required, and I may decline to participate in any or all follow-up surveys with no adverse consequences for me.

4. I understand that while information gained during the research project may be published in academic journals or books, my name and position title will not be used in relation to any of the information I have provided, unless I explicitly indicate that I am willing to be identified when quoted.

5. I understand that any personal, sensitive, or potentially incriminating information will be kept confidential so far as the law allows. Data collected from this project, including this consent form and the survey, will be kept digitally and password protected, accessible only by Luke Nguyen-Hoan. Any hard copies of data collected will be kept in a locked filing cabinet at the Australian National University.

6. I understand that although any comments I make will not be attributed to me in any publication; it is possible that others may guess the source of information, and I should avoid disclosing information which is of confidential status or which is defamatory of any person or organisation.

7. I understand that I may withdraw from the research project at any stage, without providing any reason and that this will not have any adverse consequences for me. If I choose to withdraw, any information that I have provided will not be used upon my request.

Q1. * In any publications produced as a result of this research I consent to be identified by (check one):

- My full name, position, and organisation
- My position and organisation (it is possible that you could be identified from this information)
- My organisation
- None of the above (complete anonymity)

This information will be used to track survey responses. The following information will not be used in publications unless you have indicated otherwise in the previous question. You may choose not to
publications unless you have indicated otherwise in the previous question. You may choose not to disclose this information if desired.

Q2. Full name: 

Q3. Organisation: 

Q4. Position within organisation: 

Q5. Country: 

Q6. * I consent to being asked to participate in further follow-up surveys  
   ○ Yes - please provide your email address: 
   ○ No

If you answered yes to the previous question, please provide your email address for contact purposes. Your email address will only be used to notify you of follow-up surveys if they occur, and will not be used for any other purpose, and will not be published or made available to any third parties.
Section 2. Scientific Software part 1 of 2 (Questions marked * are mandatory)

<table>
<thead>
<tr>
<th>Q7. * What programming language(s) have you commonly used for scientific applications? (Please select all that apply)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ] C</td>
</tr>
<tr>
<td>[ ] C++</td>
</tr>
<tr>
<td>[ ] Java</td>
</tr>
<tr>
<td>[ ] Fortran</td>
</tr>
<tr>
<td>[ ] Pascal</td>
</tr>
<tr>
<td>[ ] Delphi</td>
</tr>
<tr>
<td>[ ] Ruby</td>
</tr>
<tr>
<td>[ ] Perl</td>
</tr>
<tr>
<td>[ ] Python</td>
</tr>
<tr>
<td>[ ] Other (please specify)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q8. * What were the main reasons for the choice of programming language? (Please select all that apply)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ] Cross-platform compatibility</td>
</tr>
<tr>
<td>[ ] Ease of use</td>
</tr>
<tr>
<td>[ ] Developer experience</td>
</tr>
<tr>
<td>[ ] Favourite language</td>
</tr>
<tr>
<td>[ ] Features provided by language</td>
</tr>
<tr>
<td>[ ] Legacy code already written in language</td>
</tr>
<tr>
<td>[ ] Only language known</td>
</tr>
<tr>
<td>[ ] Performance</td>
</tr>
<tr>
<td>[ ] Required due to other factors (management, etc)</td>
</tr>
<tr>
<td>[ ] Other (please describe in the comments field)</td>
</tr>
</tbody>
</table>

Comments

[ ]
Q9. * Were any tools or programs such as IDEs, automated test environments, or version control software used to aid in the development of the application?

- Yes - please provide details in the comments field
- No

Comments

Q10. * If you answered yes to the previous question, please briefly explain why these tools or programs are used. If you answered no to the previous question, please briefly explain why you do not commonly use any tools or programs to aid development.

How often does the development team for the scientific software applications you work on consist of:

Q11. * Only yourself?

Q12. * A small team (2-6 people total)?

Q13. * A large team (7-12 people)?

Q14. * More than 12 people?

Q15. Are there any comments you would like to make regarding the number of people you commonly work with on scientific software projects?

How often does the intended user base for the scientific software applications you have worked on consist of:

Q16. * Only yourself?

Q17. * A small local group (less than 10 people)?

Q18. * A larger group (up to around a hundred people)?

Q19. * An international community of users?

Q20. Are there any comments you would like to make regarding the intended number of users for the scientific software applications which you have worked on?
How often does the intended user base of the applications which you have worked on consist of:

| Q21. * Only users with programming experience? | Always | Often | Sometimes | Rarely | Never |
| Q22. * Only users without programming experience? |   |   |   |   |
| Q23. * Both users with and without programming experience? |   |   |   |   |
**Scientific Software Development 2**

1 2 3 4

Section 3. Scientific Software part 2 of 2 (Questions marked * are mandatory)

<table>
<thead>
<tr>
<th>Q24. What documentation is commonly produced for the scientific software applications you have worked on? (Please select all that apply)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ] Requirements specification</td>
</tr>
<tr>
<td>[ ] Design documentation</td>
</tr>
<tr>
<td>[ ] Test results</td>
</tr>
<tr>
<td>[ ] User manuals and/or guides</td>
</tr>
<tr>
<td>[ ] Comments in the code</td>
</tr>
<tr>
<td>[ ] Theory or algorithm descriptions</td>
</tr>
<tr>
<td>[ ] Installation guide</td>
</tr>
<tr>
<td>[ ] Other (please describe in the comments field)</td>
</tr>
</tbody>
</table>

**Comments**

<table>
<thead>
<tr>
<th>Q25. * Please briefly describe the main reasons why types of documentation are or are not commonly produced for the scientific software applications you have worked on.</th>
</tr>
</thead>
</table>

**Q26.** What methods are commonly used to test, verify, or otherwise show that the scientific software applications you have worked on operate correctly? Please select all that apply. |

| [ ] Comparison with empirical (experimental or observational) data |
| [ ] Comparison with other models |
| [ ] Peer review |
| [ ] Regression tests (test against results from previous versions) |
| [ ] Unit tests (test the behaviour of individual components) |
| [ ] Integration tests (test the interaction between components) |
| [ ] Acceptance tests (test by users) |
| [ ] Verification against specified requirements |
Verification against specified requirements

☐ Verification against specified design

☐ Other (please describe in the comments field)

Comments

Q27. * Please briefly describe the main reasons why the above types of testing or verification methods are commonly performed.

Comments

Q28. Are there any other ways in which confidence of users and stakeholders is increased in the scientific software applications, so that they can feel they can safely rely on the software? If so, please briefly describe these methods:

Comments

In reference to scientific software which you have used or developed, please rate the importance of the following software quality attribute:

<table>
<thead>
<tr>
<th></th>
<th>Very Important</th>
<th>Important</th>
<th>Neither</th>
<th>Unimportant</th>
<th>Very Unimportant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q29. * Functionality (the ability to perform all required functions)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q30. * Reliability (the ability to perform with correct, consistent results)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q31. * Maintainability (the ability to be easily corrected)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q32. * Availability (the ability to be accessed and operated when needed)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q33. * Flexibility (the ability to be easily adapted to changing requirements)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q34. * Portability (the ability to be easily modified for a new software/computing environment)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q35. * Reusability (the ability to be used in multiple</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Q36. * Testability (the ability to be easily and thoroughly tested)
Q37. * Usability (the ability to be easily learned and used)
Q38. * Traceability (the ability to link the knowledge used to create the application through to the code and the output)
Q39. * Performance (the ability to run using a minimum of time and/or memory)
Q40. * Are there any comments which you would like to make regarding the quality attributes described above, particularly for those which you did not consider important?

Q41. Are there any attributes which you feel are important for scientific software which are not adequately covered by the above options?
## Section 4. Personal Details
(Questions marked * are mandatory)

<table>
<thead>
<tr>
<th>Q42.</th>
<th>Describe the scientific area in which you work in one or two sentences:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Q43. | How many years of programming experiences do you have? (answer as a  |
|      | number)                                                               |
|      |                                                                        |

<table>
<thead>
<tr>
<th>Q44.</th>
<th>What is the highest level of education you have completed?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Diploma</td>
</tr>
<tr>
<td></td>
<td>Bachelor's Degree</td>
</tr>
<tr>
<td></td>
<td>Postgraduate Diploma</td>
</tr>
<tr>
<td></td>
<td>Master's Degree</td>
</tr>
<tr>
<td></td>
<td>Doctorate</td>
</tr>
<tr>
<td></td>
<td>Other (please specify)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q45.</th>
<th>What area was your education completed in?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Q46. | If you have any other comments regarding the poll or its contents,    |
|      | please provide any feedback here:                                    |
|      |                                                                        |

« Previous  | Next »
Appendix C

DSVL Study: Ethics Approval
HUMAN RESEARCH ETHICS COMMITTEE
Application Form

Created by: u4113344
Record number: 4261
Protocol type: Expedited Ethical Review (E1)
Protocol number: 2011/094

Date entered: 28/02/2011
Ethics program type: Postgraduate
Requested start date: 01/04/2011
Requested end date: 01/10/2011

Protocol title: Scientists developing and using VDSLs

Investigators

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flint, Shayne</td>
<td>Supervisor</td>
<td>Research School of Computer Science, College of Engineering and Computer Science, ANU</td>
</tr>
<tr>
<td>Nguyen-Hoan, Luke</td>
<td>Primary investigator</td>
<td>Research School of Computer Science, College of Engineering and Computer Science, ANU</td>
</tr>
<tr>
<td>Sankaranarayana, Ramesh S</td>
<td>Supervisor</td>
<td>Research School of Computer Science, College of Engineering and Computer Science, ANU</td>
</tr>
</tbody>
</table>

Investigators Detailed

Name: Flint, Shayne  Role: Supervisor

Expertise: Shayne Flint is a Senior Lecturer in the School of Computer Science within the College of Engineering and Computer Science at the Australian National University. Shayne has broad experience in the development and application of software and system engineering methodologies in industry, government and academia. His current work is building upon this background to develop approaches software-intensive systems engineering within multi-disciplinary environments. Shayne is a Chartered Professional Engineer (Member, Engineers Australia) and a member of IEEE.
Name: Nguyen-Hoan, Luke  Role: Primary investigator

Expertise: Luke Nguyen-Hoan is a PhD Candidate in his fourth year of candidature in the School of Computer Science within the College of Engineering and Computer Science at the Australian National University. His PhD research is in the area of supporting and improving scientific software development and usage with software engineering techniques and methodologies. He has previously run a web-based survey of scientists regarding their software development habits.

Name: Sankaranarayana, Ramesh S  Role: Supervisor

Expertise: Ramesh Sankaranarayan is a Senior Lecturer in the School of Computer Science within the College of Engineering and Computer Science at the Australian National University. Ramesh's current research interests are in the area of Software-Intensive Systems Engineering and Information Retrieval. He has supervised two PhD students, namely, Mr. Tim Jones (current) and Dr. Tin Tang (completed), who required ethics approval for carrying out experiments that involved the use of human judges to evaluate the relevance and quality of search results returned by search engines.

---

External Investigators

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Departments

<table>
<thead>
<tr>
<th>Primary</th>
<th>Department</th>
<th>Faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Research School of Computer Science</td>
<td>College of Engineering and Computer Science</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Project Questions Detailed

Description of Project

Describe the research project in terms easily understood by a lay reader, using simple and non-technical language. Firstly, to investigate what kinds of visual, graphical diagrams ecological scientists draw when presented with a ecological computer simulation scenario and compare the characteristics of those diagrams with
those used by software engineers when developing computer software. Secondly, to compare and contrast using a visual diagram-based software interface (Visual Domain-Specific Modeling and Languages, or VDSM and VDSL) against a traditional fill-in-the-box software interface and determine which is considered more useful and easy to use by scientists.

Location of Data Collection

Australia Yes

Overseas No

Provide country / area where data collection will be conducted ACT, Australian National University

Aims of the Project

List the hypothesis and objectives of your research project. Hypothesis 1: Diagrams drawn by scientists to describe an ecological system are comparable to those drawn by software engineers to describe a software system. Hypothesis 2: Scientists find the visual diagram-based software interface more useful than the traditional fill-in-the-box software interface. Hypothesis 3: Scientists find the visual diagram-based software interface more easy-to-use than the traditional fill-in-the-box software interface.

Methodology

In language appropriate for a lay reader, explain why the methodological approach minimises the risk to participants. (For surveys, include justification of the sample size). Full details of the research project and the types of data collected will be explained in writing to the participants and their written consent gained prior to the research project commencing. Participants will be able to choose to end their participation at any point up during the research project and all data collected up to that point be destroyed if they choose.

Provide the survey method, a list of the questions to be asked or an indicative sample of questions. These should give a good sense of the most intrusive/sensitive areas of questioning. Participants will be questioned in a semi-structured interview, and complete a short written questionnaire where they rate their experience on a series of Likert scales. Participants will also be asked about their experience levels in the software used, ecological studies, and ecological simulation modelling.

A sample of the questions to be asked in the semi-structured interview are:
- Could you give me an overview of diagram you have drawn?
- Why did you choose to draw these elements as boxes/lines/circles/etc?
- Why did you lay out the diagram in this way?
- What were the best features of the diagram-based interface?
- What were the worst features of the diagram-based interface?
- What would you like to see improved in the diagram-based interface?

A sample of the questions to be asked in the written questionnaire are:
- Please rate the following statements from extremely unlikely to extremely likely, where
your "job" is to create a software ecological simulation:
- Using the diagram-based interface in my job would enable me to accomplish tasks more quickly.
- Using the diagram-based interface would enhance my effectiveness on the job.
- I would find it easy to get the diagram-based interface to do what I want it to do.
- It would be easy for me to become skillful at using the the diagram-based interface.

What mechanisms do the researchers intend to implement to monitor the conduct and progress of the research project? For example:
- How often will the researcher be in touch with the supervisor?
- Is data collection going as expected? If not, what will the researcher do?
- Is the recruitment process effective?
- How will the researcher monitor participants willingness to continue participation in the research project, particularly when the research is ongoing? - The researcher will meet supervisors weekly.
- As the research project has not commenced, any future problems encountered will be discussed with the supervisors to determine the most appropriate course of action.
- Written consent will be obtained from participants prior to their involvement in the research project, and participants are able to end their participation at any point during the research project.

Participants

Provide details in relation to the potential participant pool, including:

- target participant group;
- identification of potential participants;
- initial contact method, and
- recruitment method. Target participant group: Ecological scientists and ecological studies students
Identification of potential participants: By references from contacts in the field and lecturers
Initial contact method: By email or meeting in person
Recruitment method: By email or meeting in person

Proposed number of participants 20

Provide details as to why these participants have been chosen? Convenient sample of ecological scientists at the Fenner School of Environment and Society at the Australian National University.

Cultural and Social Considerations/Sensitivities

What cultural and/or social considerations/sensitivities are relevant to the participants in this research project? None.

Incentives

Will participants be paid or any incentives offered? If so, provide justification and details. Students will be given one free movie pass from either Dendy’s or Hoyts (yet to be determined) to compensate them for their time.
Benefits

What are the anticipated benefits of the research? This research is anticipated to provide insights and understanding of scientists' views of a diagram-based software engineering technique, and to gain knowledge of how the technique can be adapted or improved to cater for scientists.

To whom will the benefits flow? Software engineers working with scientists, by improving the way they present this technique to scientists, and scientists themselves, by having an improved technique for them to use.

Informed Consent

Indicate how informed consent will be obtained from participants. At least one of the following boxes MUST be ticked 'Yes'.

In writing Yes

Return of survey or questionnaire No

Orally No

Other No

If Oral Consent or Other, provide details.

Confidentiality

Describe the procedures that will be adopted to ensure confidentiality during the collection phase and in the publication of results. Results of the survey and the identities of participants will only be accessible in full to the investigators and supervisors named in this application, excepting cases where the law requires otherwise.

Data Storage Procedures

Provide an overview of the data storage procedures for the research. Include security measures and duration of storage. Physical records of data will be stored in a locked filing cabinet in a locked office of principal investigator. Digital records of data (interview recordings, etc) will be stored on the principal investigator's department computer. These records are backed up by department procedures, and is will not leave the department.

Feedback

Provide details of how the results of the research will be reported / disseminated, including the appropriate provision of results to participants. If appropriate, provide details of any planned debriefing of participants. Results of the research may be published in an appropriate academic conference. Participants will be provided with a link to the principal investigator's website where publications will be referenced.
HUMAN RESEARCH ETHICS COMMITTEE
Application Form

Supporting Documentation

Please ensure electronic copies of any supporting documentation have been uploaded in the documents tab of the relevant protocol.

Has this work been approved by another Human Research Ethics Committee (HREC)? No

If yes, please give the name of the approving HREC.

Funding

Is this research supported by external funding? No

Provide the name/s of the external sources of funding. Please include grant number/s if available.

Is the research conducted under the terms of a contract of consultancy agreement between the ANU and the funding source? No

Describe all the contractual rights of the funding source that relate to the ethical consideration of the research.
### High Risk One Summary

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is this a clinical trial?</td>
<td>No</td>
</tr>
<tr>
<td>Does this research involve the intentional recruitment or issues involving Aboriginal and / or Torres Strait Islander Peoples?</td>
<td>No</td>
</tr>
</tbody>
</table>

### High Risk Two Summary

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does this research involve Human Genetics?</td>
<td>No</td>
</tr>
<tr>
<td>Does this research involve Human Stem Cells?</td>
<td>No</td>
</tr>
<tr>
<td>Does this research involve Women who are pregnant and the Human Foetus?</td>
<td>No</td>
</tr>
<tr>
<td>Does the research involve people highly dependent on medical care who may be unable to give consent?</td>
<td>No</td>
</tr>
<tr>
<td>Does the research involve people with a cognitive impairment, an intellectual disability or a mental illness?</td>
<td>No</td>
</tr>
<tr>
<td>Does this research involve an intention to study or expose or is likely to discover illegal activity?</td>
<td>No</td>
</tr>
</tbody>
</table>

### Expedited Questions Summary

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Third Party Identification</td>
<td>No</td>
</tr>
<tr>
<td>Children or Young People</td>
<td>No</td>
</tr>
<tr>
<td>Dependent or Unequal Relationship</td>
<td>No</td>
</tr>
<tr>
<td>Membership of a Group, or Related Issues</td>
<td>No</td>
</tr>
<tr>
<td>Physical Harm</td>
<td>No</td>
</tr>
<tr>
<td>Psychological Harm (includes Devaluation of Personal Worth)</td>
<td>No</td>
</tr>
<tr>
<td>Social Harm</td>
<td>No</td>
</tr>
<tr>
<td>Economic Harm</td>
<td>No</td>
</tr>
<tr>
<td>Legal Harm</td>
<td>No</td>
</tr>
<tr>
<td>Covert Observation</td>
<td>No</td>
</tr>
<tr>
<td>Deception</td>
<td>No</td>
</tr>
<tr>
<td>Sensitive Personal Information</td>
<td>No</td>
</tr>
<tr>
<td>Overseas Research</td>
<td>No</td>
</tr>
<tr>
<td>Collection, use or disclosure of personal information WITHOUT the consent of the participant</td>
<td>No</td>
</tr>
</tbody>
</table>
Appendix D

DSVL Study: Information Sheet
Creating Ecological Simulations using Visual Domain-Specific Modeling

INFORMATION SHEET

This project has been designed to examine how scientists view and use Visual Domain-Specific Modeling, a technique used in software engineering to describe systems and scenarios in a visual, diagram-based manner. More specifically, this research consists of two sub-components:

1. to investigate the characteristics of visual, graphical diagrams that ecological scientists draw when presented with an ecological computer simulation scenario; and
2. to compare and contrast the experiences of ecological scientists creating an ecological software simulation using a visual diagram-based software interface against a traditional fill-in-the-box form-based software interface.

Why this research is being carried out

This research is being performed in order to gain an understanding of how ecological scientists interact with and use visual diagram-based tools compared with more traditional software interfaces. The information gained from this study will be used to identify how the interface between scientists and software systems can be improved, and in what ways Visual Domain-Specific Modeling can be adapted or improved for use by scientists.

What this research involves

This research will involve the following activities:

1. Read a one-page summary of an ecological scenario;
2. Draw a representation of the ecological scenario;
3. Using both a visual diagram-based software interface and a traditional fill-in-the-box form-based software interface to create a software model of the ecological scenario;
4. Complete a short survey of your experiences during the research project; and
5. A short semi-structured interview where you will be asked questions relating to both your drawing and your experiences using the software interfaces

These activities are anticipated to take no more than two (2) hours in total.

The interviewer will take notes of the semi-structured interview, and participants' names, job titles, and place of employment will not be revealed in subsequent
research publications and reports without explicit consent. No quotes or attributed opinions will be used without the explicit permission of the interviewee. Participants will also be asked about previous experience in ecological studies, modeling, and software simulations, and this information will be used to categorise participants. Given permission, interviews will also be recorded to back up any written notes. You may refuse to answer certain questions, withdraw from the interview at any time, or request that material not be used.

Participation in this project is purely voluntary and there will be no adverse consequences if you decide not to participate. You may withdraw from participation in the project at any time, and you are not required to provide any reason for doing so. If you choose to withdraw, any information that you have previously provided will not be used if so requested.

The results of this research will be reported in a doctoral thesis and may be published in academic journals or books. Once published, the results of this research will be made available upon request.

Potential risks in participation

The survey has not been designed to elicit any information that is confidential, sensitive, or potentially incriminating in nature. Answering all the questions is not required, and there are no consequences to the participant in not answering any given question, although the more complete answers are the more useful they are to the researcher. Should you have any further concerns about confidentiality and anonymity, please raise the issue prior to or immediately after the project so that assurances can be made that specific information is not reported or published.
Contact names and phone numbers

If you have any questions, comments, or complaints about this research, please do not hesitate to contact:

Luke Nguyen-Hoan
Research School of Computer Science
College of Engineering and Computer Science
The Australian National University
Phone: +61 2 612 59663 (W) or +61 422 265 826 (M)
Email: luke.nguyen-hoan@anu.edu.au

or

Dr Shayne Flint
Research School of Computer Science
College of Engineering and Computer Science
The Australian National University
Phone: +61 2 612 58183
Email: shayne.flint@anu.edu.au

If you have any concerns regarding the way the research was conducted you can also contact the ANU Human Research Ethics Committee:

Human Ethics Officer
Human Research Ethics Committee
The Australian National University
Phone: 02 6125 3427
Email: human.ethics.officer@anu.edu.au
Appendix E

DSVL Study: Consent Form
CONSENT FORM

1. I have read the information sheet for this project and understand its contents. I have had the nature and purpose of the research project, so far as it affects me, fully explained to my satisfaction by the research worker. My consent is freely given.

2. I understand that if I agree to participate in the research project I will be asked to perform a series of tasks, complete a written survey, and be questioned in a recorded semi-structured interview. These activities will take no more than two (2) hours to complete.

3. I understand that while information gained during the research project may be published in academic journals or books, my name will not be used in relation to any of the information I have provided, unless I explicitly indicate that I am willing to be identified when quoted.

4. I understand that any personal, sensitive, or potentially incriminating information will be kept confidential so far as the law allows. Data collected from this project, including this consent form and the survey, will be kept digitally and password protected, accessible only by Luke Nguyen-Haan. Any hard copies of data collected will be kept in a locked filing cabinet at the Australian National University.

5. I understand that although any comments I make will not be attributed to me in any publication, it is possible that others may guess the source of information, and I should avoid disclosing information which is of confidential status or which is defamatory of any person or organisation.

6. I understand that I may withdraw from the research project at any stage, without providing any reason and that this will not have any adverse consequences for me. If I choose to withdraw, any information that I have provided will not be used upon my request.
7. In any publications produced as a result of this research, I consent to be identified by (check one):
   [ ] My full name, position, and place of employment
   [ ] My position and organisation (if you tick this box it is possible that you could be identified)
   [ ] My organisation
   [ ] None of the above (complete anonymity)

8. In any publications produced as a result of this research, I consent to be directly quoted (check one):
   [ ] With attribution, subject to the above
   [ ] Without attribution (anonymously)
   [ ] I do not consent to be directly quoted

9. In order to back up written notes, I consent that an audio recording of the interview can be taken (check one):
   [ ] Yes
   [ ] No

I agree to the above statements and consent to participating in the study of Creating Ecological Simulations using Visual Domain-Specific Modeling.

Name (please print):

Signature:

Date:
Appendix F

Environmental Scenario Description
**Scenario: European Larch**

Feel free to ask any questions you may have regarding this scenario.

This scenario is adapted from Albert et al, 2008. Some segments are directly quoted from the original publication.

In many places in Europe, trees are currently far below their theoretical altitudinal limit because of past land use. However, under the current crisis affecting mountain agriculture, trees are likely to recolonise the subalpine belt. This scenario aims to analyse the factors determining tree dynamics at the subalpine ecotone and to evaluate the impact of land-use change on landscape vegetation patterns.

We would like to perform a landscape scale analysis of the colonisation of European larch (*Larix decidua* Mill.) in a subalpine region which is currently mountain grasslands due to past land use. Farming practices prevent trees remaining in this region. However, with land abandonment due to farmers leaving their businesses for whatever reasons, trees may be able to return.

Given knowledge of the climatically suitable habitats for European larch, we will compare the ability of larch to colonise those grasslands under three land-use cases, considering also the dispersal ability of larch and its ability to compete against the local vegetation.

Existing grassland species present in the area of interest can be categorised into four types (named after the archetype species in each cluster):

1. "Bromus" - medium-sized, disturbance-tolerant species
2. "Dactylis" - tall, acquisitive species
3. "Festuca" - tall, competitive and conservative species
4. "Sesleria" - small, stress-tolerant species

Critical autecological parameters of interest for larch are:

1. With vs. without long-distance seed dispersal ability
2. Shade-tolerant vs. intolerant juveniles
3. High vs low resource-uptake efficiency

The grasslands will be mowed early, mowed late, or abandoned. The effects of these land-use cases are:

1. Mowed early: where sections of grassland are mown "early" during summer, preventing larch establishment and preventing "Festuca" type species from producing seeds.
2. Mowed late: where sections of grassland are mown "late" at the end of each summer, preventing larch establishment.
3. Abandoned, where no land management or land-use occurs.

---

Appendix G

DSVL Study: Diagrams By Participants
Appendix G. DSL Study: Diagrams by Participants

Theoretical tree line
Subalpine belt - mountain grasslands because of human use

- Grassland types,
  (plots within Subalpine Belt).

L_1 / L_2 = seed dispersal
L_3 / L_4 = shade tolerance for juveniles
L_5 / L_6 = heterogeneity efficiency

If grass recovers quickly, no room for larch.

F/D better locations for larch from B/S.

Early or no early mowing should enable larch to establish.
D early mowing only possibility.
null
Simple:
- Grassland
  - Treatment on absence of treatment
- Forest

Complex:
- Late moving
  - Mixed ground
    - Early moving
      - Treatment: P.B.S.
      - Abandonment
- Mixed ground with trees

* = Grass
¥ = Tree
APPENDIX G. DSVL STUDY: DIAGRAMS BY PARTICIPANTS

#8

![Diagram with labels and symbols]
Sub-alpine belt
'Abandoned grazing'

Aim: Test 1: Landscape ecology
Species to be re-established

Current area of forest
'European larch'

Table:

<table>
<thead>
<tr>
<th>Species</th>
<th>Test 1</th>
<th>Test 2</th>
<th>Test 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shade</td>
<td>warm x wet</td>
<td>shade</td>
<td>warm x dry</td>
</tr>
<tr>
<td>Param 1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
APPENDIX G. DSVL STUDY: DIAGRAMS BY PARTICIPANTS

Diagram 1:

Legend:
- Yes
- No
- Unknown

Legend:
- Low
- Medium
- High

Legend:
- Factor X
- Factor Y
- Factor Z

Legend:
- Group A
- Group B
- Group C
APPENDIX G. DSVL STUDY: DIAGRAMS BY PARTICIPANTS
Appendix H

DSVL Study: Questionnaire
Questionnaire – Preliminary

Although answering all questions is not compulsory, the more complete your answers are the more useful your participation will be to the research project.

Personal Details

Important: Providing your personal information is entirely voluntary. Your full name, position, and place of employment will not be published unless you have provided consent on the original consent form.

Name: ____________________________________________

Position: __________________________________________

Organisation: ______________________________________

Past Experience

Please indicate which of the following categories best describes your position? (This is used to categorise participants into employment groups).

- Undergraduate
- Postgraduate
- Academic
- General Staff
- Other (please describe)

Do you perform ecological modelling?  
Yes  No

If yes, how many years of ecological modelling experience do you have?  
0-5  6-10  11-15  16-20  21-25  26-30  31-35  36-40  41+

Do you program ecological software simulations?  
Yes  No

If yes, have you used a visual or diagram-based software tool before?  
Yes  No

If yes, please list what visual or diagram-based tools you have used:

Have you heard of the Landscape MOdelling Shell (LAMOS) before?  
Yes  No

If yes, have you used LAMOS before?  
Yes  No
Questionnaire – Form-based Interface

For each statement, please circle the response that best fits your opinion.

I used the form-based interface:

<table>
<thead>
<tr>
<th></th>
<th>First</th>
<th>Second</th>
</tr>
</thead>
</table>

With respect to ecological modelling:

1. Using the form-based interface would enable me to accomplish tasks more quickly.

<table>
<thead>
<tr>
<th></th>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

2. Using the form-based interface would improve my performance.

<table>
<thead>
<tr>
<th></th>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

3. Using the form-based interface would increase my productivity.

<table>
<thead>
<tr>
<th></th>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

4. Using the form-based interface would enhance my effectiveness.

<table>
<thead>
<tr>
<th></th>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

5. Using the form-based interface would make modelling easier.

<table>
<thead>
<tr>
<th></th>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

6. I would find the form-based interface useful.

<table>
<thead>
<tr>
<th></th>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>
7. Learning to operate the form-based interface would be easy for me.

<table>
<thead>
<tr>
<th></th>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

8. I would find it easy to get the form-based interface to do what I want it to do.

<table>
<thead>
<tr>
<th></th>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

9. My interaction with the form-based interface would be clear and understandable.

<table>
<thead>
<tr>
<th></th>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

10. I would find the form-based interface to be flexible to interact with.

<table>
<thead>
<tr>
<th></th>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

11. It would be easy for me to become skilful at using the form-based interface.

<table>
<thead>
<tr>
<th></th>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

12. I would find the form-based interface easy to use.

<table>
<thead>
<tr>
<th></th>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>
13. I would easily understand the scenario when using the form-based interface.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

14. The form-based interface would give me a clear view of the scenario.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

15. The form-based interface would allow me to explore the scenario easily.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

16. I would be able to grasp the scenario structure easily using the form-based interface.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

17. Using the form-based interface would make it easy to communicate the scenario with others.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

18. I would easily understand a different scenario if it were presented to me using the form-based interface.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>
Questionnaire – Diagram-based Interface

For each statement, please circle the response that best fits your opinion.

I used the diagram-based interface:

<table>
<thead>
<tr>
<th></th>
<th>First</th>
<th>Second</th>
</tr>
</thead>
</table>

With respect to ecological modelling:

1. Using the diagram-based interface would enable me to accomplish tasks more quickly.

<table>
<thead>
<tr>
<th></th>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

2. Using the diagram-based interface would improve my performance.

<table>
<thead>
<tr>
<th></th>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

3. Using the diagram-based interface would increase my productivity.

<table>
<thead>
<tr>
<th></th>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

4. Using the diagram-based interface would enhance my effectiveness

<table>
<thead>
<tr>
<th></th>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

5. Using the diagram-based interface would make modelling easier.

<table>
<thead>
<tr>
<th></th>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

6. I would find the diagram-based interface useful.

<table>
<thead>
<tr>
<th></th>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>
7. Learning to operate the diagram-based interface would be easy for me.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

8. I would find it easy to get the diagram-based interface to do what I want it to do.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

9. My interaction with the diagram-based interface would be clear and understandable.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

10. I would find the diagram-based interface to be flexible to interact with.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

11. It would be easy for me to become skilful at using the diagram-based interface.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

12. I would find the diagram-based interface easy to use.

| Extremely Unlikely | Quite Unlikely | Slightly Unlikely | Neither | Slightly Likely | Quite Likely | Extremely Likely |
13. I would easily understand the scenario when using the diagram-based interface.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

14. The diagram-based interface would give me a clear view of the scenario.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

15. The diagram-based interface would allow me to explore the scenario easily.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

16. I would be able to grasp the scenario structure easily using the diagram-based interface.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

17. Using the diagram-based interface would make it easy to communicate the scenario with others.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

18. I would easily understand a different scenario if it were presented to me using the diagram-based interface.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>
Appendix I

Extended DSVL Study:
Information Sheet
Recording Context & Provenance using Visual Domain-Specific Modeling

INFORMATION SHEET

This project has been designed to examine the use of Visual Domain-Specific Modeling, a technique used in software engineering to describe systems and scenarios in a visual, diagram-based manner, in recording context and provenance information in environmental modelling. More specifically, this research consists of two sub-components:

1. to investigate how scientists currently record context and provenance information
2. to investigate the perceptions of scientists towards using a Visual Domain-Specific Modeling approach to recording context and provenance information

Why this research is being carried out

This research is being performed in order to gain an understanding of whether environmental scientists would find a Visual Domain-Specific Modeling approach to recording context and provenance useful and easy to use.

What this research involves

This research will involve the following activities:

1. A short semi-structured interview where you will be asked questions relating to how you currently record context and provenance information;
2. Reading a one-page summary of a sample scenario;
3. If appropriate, sketch or draw an example of how you would currently record and represent the context and provenance information;
4. Use a Visual-Domain-Specific Modeling approach to record and represent the context and provenance information;
5. A short survey of your experiences during the research project; and
6. A short semi-structured interview where you will be asked questions to explore and confirm your survey answers.

These activities are anticipated to take no more than one (1) hour in total.
The interviewer will take notes of the semi-structured interview, and participants' names, job titles, and place of employment will not be revealed in subsequent research publications and reports without explicit consent. No quotes or attributed opinions will be used without the explicit permission of the interviewee. Participants will also be asked about previous experience in environmental studies, modeling, and software tools, and this information will be used to categorise participants. Given permission, interviews will also be recorded to back up any written notes. You may refuse to answer certain questions, withdraw from the interview at any time, or request that material not be used.

Participation in this project is purely voluntary and there will be no adverse consequences if you decide not to participate. You may withdraw from participation in the project at any time, and you are not required to provide any reason for doing so. If you choose to withdraw, any information that you have previously provided will not be used if so requested.

The results of this research will be reported in a doctoral thesis and may be published in academic journals or books. Once published, the results of this research will be made available upon request.

Potential risks in participation

The survey has not been designed to elicit any information that is confidential, sensitive, or potentially incriminating in nature. Answering all the questions is not required, and there are no consequences to the participant in not answering any given question, although the more complete answers are the more useful they are to the researcher. Should you have any further concerns about confidentiality and anonymity, please raise the issue prior to or immediately after the project so that assurances can be made that specific information is not reported or published.
Contact names and phone numbers

If you have any questions, comments, or complaints about this research, please do not hesitate to contact:

Luke Nguyen-Hoan
Research School of Computer Science
College of Engineering and Computer Science
The Australian National University
Phone: +61 2 612 59663 (W) or +61 422 265 826 (M)
Email: luke.nguyen-hoan@anu.edu.au

or

Dr Shayne Flint
Research School of Computer Science
College of Engineering and Computer Science
The Australian National University
Phone: +61 2 612 58183
Email: shayne.flint@anu.edu.au

If you have any concerns regarding the way the research was conducted you can also contact the ANU Human Research Ethics Committee:

Human Ethics Officer
Human Research Ethics Committee
The Australian National University
Phone: 02 6125 3427
Email: human.ethics.officer@anu.edu.au
Appendix J

Extended DSVL Study: Consent Form
CONSENT FORM

1. I have read the information sheet for this project and understand its contents. I have had the nature and purpose of the research project, so far as it affects me, fully explained to my satisfaction by the research worker. My consent is freely given.

2. I understand that if I agree to participate in the research project I will be asked to perform a series of tasks, complete a written survey, and be questioned in a recorded semi-structured interview. These activities will take no more than one (1) hour to complete.

3. I understand that while information gained during the research project may be published in academic journals or books, my name will not be used in relation to any of the information I have provided, unless I explicitly indicate that I am willing to be identified when quoted.

4. I understand that any personal, sensitive, or potentially incriminating information will be kept confidential so far as the law allows. Data collected from this project, including this consent form and the survey, will be kept digitally and password protected, accessible only by Luke Nguyen-Haan. Any hard copies of data collected will be kept in a locked filing cabinet at the Australian National University.

5. I understand that although any comments I make will not be attributed to me in any publication, it is possible that others may guess the source of information, and I should avoid disclosing information which is of confidential status or which is defamatory of any person or organisation.

6. I understand that I may withdraw from the research project at any stage, without providing any reason and that this will not have any adverse consequences for me. If I choose to withdraw, any information that I have provided will not be used upon my request.
7. In any publications produced as a result of this research, I consent to be identified by (check one):
   [ ] My full name, position, and place of employment
   [ ] My position and organisation (if you tick this box it is possible that you could be identified)
   [ ] My organisation
   [ ] None of the above (complete anonymity)

8. In any publications produced as a result of this research, I consent to be directly quoted (check one):
   [ ] With attribution, subject to the above
   [ ] Without attribution (anonymously)
   [ ] I do not consent to be directly quoted

9. In order to back up written notes, I consent that an audio recording of the session can be taken (check one):
   [ ] Yes
   [ ] No

I agree to the above statements and consent to participating in the study of Recording Context & Provenance using Visual Domain-Specific Modeling.

Name (please print):

Signature:

Date:
Appendix K

Extended DSVL Study: Questionnaire
Questionnaire – Preliminary

Although answering all questions is not compulsory, the more complete your answers are the more useful your participation will be to the research project.

Personal Details

Important: Providing your personal information is entirely voluntary. Your full name, position, and place of employment will not be published unless you have provided consent on the original consent form.

Name: ____________________________________________

Position: __________________________________________

Organisation: ________________________________________
Past Experience

Please indicate which of the following categories best describes your position? (This is used to categorise participants into employment groups).

- Undergraduate
- Postgraduate
- Academic
- General Staff
- Other (please describe)

How many years of environmental modelling experience do you have?

<table>
<thead>
<tr>
<th>No experience</th>
<th>0-5</th>
<th>6-10</th>
<th>11-15</th>
<th>16-20</th>
<th>21-25</th>
<th>26-30</th>
<th>31-35</th>
<th>36-40</th>
<th>41+</th>
</tr>
</thead>
</table>

Have you used visual or diagram-based software tools before?

- Yes - please list below
- No

Please list visual or diagram-based software tools which you have used:
Questionnaire – Diagram-based tool

For each statement, please circle the response that best fits your opinion.

With respect to environmental modelling:

1. Using the diagram-based tool would enable me to accomplish tasks more quickly.
   
<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

2. Using the diagram-based tool would improve my performance.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

3. Using the diagram-based tool would increase my productivity.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

4. Using the diagram-based tool would enhance my effectiveness

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

5. Using the diagram-based tool would make modelling easier.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

6. I would find the diagram-based tool useful.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>
With respect to environmental modelling:

7. Learning to operate the diagram-based tool would be easy for me.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

8. I would find it easy to get the diagram-based tool to do what I want it to do.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

9. My interaction with the diagram-based tool would be clear and understandable.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

10. I would find the diagram-based tool to be flexible to interact with.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

11. It would be easy for me to become skilful at using the diagram-based tool.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

12. I would find the diagram-based tool easy to use.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>
With respect to environmental modelling:

13. I would easily understand the experiment when using the diagram-based tool.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

14. The diagram-based tool would give me a clear view of the experiment.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

15. The diagram-based tool would allow me to explore the experiment easily.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

16. I would be able to grasp the experiment structure easily using the diagram-based tool.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

17. Using the diagram-based tool would make it easy to communicate the experiment with others.

<table>
<thead>
<tr>
<th>Extremely Unlikely</th>
<th>Quite Unlikely</th>
<th>Slightly Unlikely</th>
<th>Neither</th>
<th>Slightly Likely</th>
<th>Quite Likely</th>
<th>Extremely Likely</th>
</tr>
</thead>
</table>

18. I would easily understand a different experiment if it were presented to me using the diagram-based tool.

| Extremely Unlikely | Quite Unlikely | Slightly Unlikely | Neither | Slightly Likely | Quite Likely | Extremely Likely |
Bibliography


