

OMNISPECTIVE ANALYSIS AND REASONING

AN EPISTEMIC APPROACH TO SCIENTIFIC WORKFLOWS

SRINIVAS CHEMBOLI



Australian
National
University

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

September 2012



This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/>.

Copyright © 2012 Srinivas Chemboli
version 12.09.04

DECLARATION

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

Srinivas Chemboli

04 September 2012

[S]cience includes any approach that is open to reason, to rational discussion, investigation, skepticism, to critical thinking, to questioning. . . I wouldn't say you have to put on a white coat and go into a laboratory in order to [pursue science].

Dawkins [2012]

ACKNOWLEDGMENTS

The research presented in this dissertation could not have been done without the help of many colleagues and friends who supported me in many ways during all the years. Only the names of some of them can be listed here.

First and foremost, I would like to thank my supervisor, Clive Boughton, for his unstinting support and encouragement. Thank you, Clive, for skillfully ensuring the right mix of ‘supervision’ and ‘independent latitude’ in research. I have learnt much these past years, and that could not have happened without your guidance.

I have benefited greatly from the valuable feedback and suggestions of my advisors, Shayne Flint and Ramesh Sankaranarayana. Thank you Shayne, for the informed critique and many hours of brainstorming and discussion. I am grateful to Ramesh for steering me towards ‘scientific workflows.’

I am also thankful to Elisa Baniassad, chair of my supervisory panel — although Elisa came on board at a later stage of my study, her advice and insights have been very helpful in shaping the research.

Over the past several years, I have had the great pleasure of interacting with an incredible group of people in the School of Computer Science.

In particular, I would like to thank Henry Gardner for taking an active interest in my research and offering valuable advice.

I am also thankful to Lynette Johns–Boast, Chris Johnson, Clem Baker–Finch and Alexei Khorev for giving me the opportunity to learn to teach. The skills acquired from this experience have been very useful in my research. I would like to thank Tom Worthington, Malcolm Newey and Jay Larson for numerous helpful and enlightening discussions about conducting and presenting research.

To Julie Arnold, Chelsea Holton, Fiona Quinlan, Bethany Flanders, Suzy Andrew, Debbie Pioch, Jill Mayo, Marie Katselas, Sue van Haeften, Jonathan Peters, Paul Melloy and Jadon Radcliffe: A Big Thank You for cheerfully handling all administrative niggles. Special thanks to Bob Edwards, Hugh Fisher, Steven Hanley and in particular to James Fellows, for technical support and wizardry.

During my PhD, I also had the opportunity to work as an educational technologist in the Division of Information at the ANU. I would like to thank Karen Visser, Jenny Edwards, Marina Lobastov, Peter Yates, Jaymie Parker, Sam Primrose, Grazia Scotellaro, Candida Spence, Hans Joerg–Kraus, Kim Blackmore and Lauren Thompson — it was a privilege to work with you.

In addition, I also participated in the Pinnacle Teaching Program, the Graduate Teaching Program and the Foundations of University Teaching and Learning course. I am thankful to Trevor Vickers and Beth Beckmann for their support in these programs.

It would be remiss of me not to express my thanks to Mark Drechsler for insights of Moodle and learning management systems.

I would also like to thank Deborah Veness for her comments and suggestions on the application of OAR to course design and for encouraging me to present the work at ascilite.

I am thankful to Tony Karrys, Gina Denman, Kaori Oikawa and Walter Sauer for their assistance and kindness.

This research has been supported by the Australian National University, and the Commonwealth of Australia, through the Cooperative Research Centre for Advanced Automotive Technology. I would like to thank Kate Neely for her support and assistance in coordinating with the AutoCRC.

I would like to thank the many people with whom I have shared my time at CECS: Agung Fatwanto, Normi Abu Bakar, Zoe Brain, Luke Nguyen–Hoan, Alvin Teh, Amir Hadad, Sukanya Manna, Josh Milthorpe, Jaison Mulerikkal, Derek Wang and Ian Wood. You have been excellent company. I must particularly thank my officemate and friend Ziyad Alshaikh. Thank you for the sparkling wit and interesting discussions — no research ideas were too outlandish for conversation!

Special thanks to:

Aditi Barthwal and Kate Sullivan, for your friendship, support and encouragement.

Josephine Wright, for your optimism and friendly advice.

Ganesh Venkateswara, for taking time out and providing thoughtful insights and feedback.

Finally, I would like to thank my family for their continuous support and encouragement. Without you this work would never have been possible.

ACKNOWLEDGMENTS OF PERMISSIONS

I want to acknowledge the permissions of various copyright holders.

The Workflow Management Coalition kindly permitted me to use the following material:

From D. Hollingsworth, "The Workflow Reference Model", Document Number TC00-1003, The Workflow Management Coalition, Hampshire, UK : 1995 <http://www.wfmc.org/standards/docs/tc003v11.pdf>,

Figure 6 (Figure 2.1 in the thesis, redrawn for style and appearance)

Dr Shayne Flint of the Australian National University kindly permitted me to use the following material:

From S. Flint, "Rethinking Systems Thinking", Proceedings of the 14th ANZSYS Australia New Zealand Systems Society Conference, Perth, Australia : 2008 <http://www.anzsys.org/anzsys08/papers/Shayne%20Flint%20Rethinking%20Systems%20Thinking.pdf>

Figure 2 (Figure 3.1 in the thesis, redrawn for style and appearance)

The International Society for the Systems Sciences kindly permitted me to use the following material:

From J.N. Warfield, "The Domain of Science Model: Evolution and Design", Proceedings of the Society for General Systems Research, Salinas, CA: Intersystems, 1986, H46-H59,

Figure 1 (Figure 3.2 in the thesis, redrawn for style and appearance)

Lynette Johns-Boast of the Australian National University kindly permitted me to use the following material:

"A Model of Curriculum Design", Personal communication : 2010,

(Figure 5.1 in the thesis, redrawn for style and appearance)

ABSTRACT

This thesis presents the conceptualization, formulation, development and demonstration of Omnispersive Analysis and Reasoning (OAR), an epistemic framework for managing intellectual concerns in scientific workflows.

Although scientific workflows are extensively used to support the management of experimental and computational research, intellectual concerns are not adequately handled in current practice owing to the focus on low-level implementation details, limited context support, issues in developing shared semantics across disciplines and lack of support for verification and validation of the underlying science of the workflow. The management of intellectual concerns in scientific workflows can be improved by developing a framework for providing a layer of abstraction to lift focus from low-level implementation details, adding context as a workflow parameter, introducing localized ontologies and abstracting and mapping intellectual concerns in the research-domain to workflow specification and execution semantics.

Following an examination of typical definitions of scientific workflow offered in literature, the Scientific Method is applied to develop an enhanced definition of a scientific workflow. This definition, which extends the scope of ordered analysis and investigation to a generic problem scenario, is utilized in the OAR framework. The design of OAR is modular like the Domain of Science Model (DoSM). The structure and working of OAR incorporate the evolving nature of science, hierarchy of conceptualization, omnisppection, and the logical processes of analysis, reasoning and abstraction. These form the Foundation and Theory of OAR. Abstracting concerns in terms of unit knowledge entities (ukes) and groups of ukes (recipes), use of context to identify relation between recipes, the management of recipes in shelves, and the processes of concern refinement and context refinement constitute the Methodology.

A comprehensive and simple example of the application of OAR to the abstraction, analysis, formulation and orchestration of a scientific workflow at different levels of granularity is provided by applying it to the problem of origami paper folding. The use of OAR in capturing the rationale of design decisions and mapping them to desired outcomes is demonstrated by applying OAR for contextualizing course design. Another example illustrates the use of OAR in the analysis, understanding and management of complex systems. Localized ontologies enable the exposure of side-effects and emergent behavior in

large-scale systems due to the choice of any particular solution specification. These examples constitute a first step in building the Applications block of OAR. While OAR may be manually applied even to large-scale problems, it is expedient to avail of tool support. Soma — a simple and illustrative tool prototype is developed to indicate directions for a reference tool implementation.

The thesis concludes with a consideration of ideas for future work. The contribution in this thesis corresponds to an instance of the DoSM for scientific workflow management. The OAR framework has great potential for further development as a well-formed Science of Workflows.

PUBLICATIONS

Some ideas and figures have appeared previously in the following presentations and publications:

Chemboli, S. and Boughton, C. [2012a]. "Managing Large and Complex Systems with Omnispective Analysis and Reasoning." In: *Proceedings of SETE APCOSE 2012*. Brisbane, Australia. URL: <http://hdl.handle.net/1885/9009>

Chemboli, S. and Boughton, C. [2012b]. "Omnispective Analysis and Reasoning: A Framework for Managing Intellectual Concerns in Scientific Workflows." In: *Proceedings of the 5th India Software Engineering Conference*. Kanpur, India, pp. 143–146. DOI: [10.1145/2134254.2134279](https://doi.org/10.1145/2134254.2134279)

Chemboli, S. and Boughton, C. [2011]. "Contextual Course Design with Omnispective Analysis and Reasoning." In: *Changing Demands, Changing Directions. Proceedings ascilite*. Ed. by Williams, G. et al. Hobart, pp. 210–219. URL: <http://www.leishman-associates.com.au/ascilite2011/downloads/papers/Chemboli-full.pdf>

Chemboli, S. [Oct. 2010a]. *Contextualizing learning outcomes and course design in Moodle*. Presented at Moodleposium AU 2010. Canberra, Australia. URL: <http://hdl.handle.net/1885/9279>

Chemboli, S., Kane, L., and Johns-Boast, L. [July 2010]. *Translating Learning Outcomes in Moodle*. Presented at Moodlemoot AU 2010. Melbourne, Victoria, Australia. URL: <http://ubuntuone.com/4RBjlozHyEyITuy21aDCfe>

Chemboli, S. [Feb. 2010b]. *Omnispective Analysis and Reasoning: An epistemic approach to scientific workflows*. Presented at the CECS Seminar Series, Australian National University. Canberra, Australia. URL: <http://cecs.anu.edu.au/seminars/more/SID/2503>

CONTENTS

ACKNOWLEDGMENTS [vii](#)

ABSTRACT [xi](#)

PUBLICATIONS [xiii](#)

CONTENTS [xv](#)

LIST OF FIGURES [xix](#)

LIST OF TABLES [xxi](#)

I Introduction [1](#)

1 OVERVIEW [3](#)

- 1.1 Initial Motivation and Research Aim [4](#)
- 1.2 Research Design [4](#)
- 1.3 Thesis Scope [5](#)
- 1.4 Thesis Structure [5](#)
 - 1.4.1 [Part I: Introduction](#) [5](#)
 - 1.4.2 [Part II: Omnispersive Analysis and Reasoning](#) [7](#)
 - 1.4.3 [Part III: Proof of Concept](#) [7](#)
 - 1.4.4 [Part IV: Conclusion](#) [7](#)
- 1.5 Main Contributions [7](#)

2 BACKGROUND [9](#)

- 2.1 Workflows [10](#)
 - 2.1.1 Scientific Database Systems [10](#)
 - 2.1.2 Unstructured Activities [11](#)
 - 2.1.3 Dynamic Process Composition [13](#)
 - 2.1.4 Distributed and Decentralized Processes [13](#)
 - 2.1.5 The Workflow Reference Model [15](#)
 - 2.1.6 Participatory Analysis [17](#)
 - 2.1.7 Emergence of Scientific Workflows [18](#)
 - 2.1.8 Scientific Workflows [20](#)
- 2.2 Issues in Scientific Workflow Management [23](#)

2.2.1	Focus on Low-level Detail	23
2.2.2	Limited Context Support	26
2.2.3	Inadequate Management of Intellectual Concerns	28
2.3	Impact of the Above Issues	29
2.3.1	Observation 1	30
2.3.2	Observation 2	30
2.3.3	Observation 3	30
2.4	Conjecture	31
2.5	Summary and Conclusion	32

II Omniscient Analysis and Reasoning 33

3	OMNISPECTIVE ANALYSIS AND REASONING	35
3.1	The Nature of Science	36
3.2	Fixation of Belief	38
3.3	Universal Priors to Science	39
3.4	Law of Triadic Compatibility	40
3.5	Hierarchy of Conceptualization	41
3.6	Domain of Science Model	42
3.7	Expanding Scale and Complexity of Scientific Work	44
3.8	Intellectual Concerns in Scientific Work	44
3.9	Defining Scientific Workflows	45
3.10	'Reforming' Scientific Workflow Management	48
3.10.1	Managing Intellectual Concerns	49
3.10.2	Dealing with Inadequate Context Support	51
3.10.3	Inadequate Support for Verification and Validation	51
3.11	Theoretical Foundations of OAR	52
3.11.1	Omniscient Analysis	53
3.11.2	Lifting Focus from Low-level Details	54
3.11.3	Defining Context and Adding Context Support	56
3.11.4	Localized Ontologies	58
3.11.5	Epistemological Basis	60
3.12	Overview of the OAR Framework	60
3.13	Prototypes, Archetypes and Constraints	62
3.14	Concern Refinement	63
3.15	'Bootstrapping' External Shelves	65
3.16	Context Refinement	66
3.17	Constructed and Organic Solution Specifications	70
3.18	Rationale for the Structure of OAR	71

3.19	Nature of the OAR Framework	73
3.20	Summary and Conclusion	74

III Proof of Concept 77

4	ORIGAMI FOLDING WORKFLOW	81
4.1	Paper Folding as a Scientific Workflow	82
4.2	Folding the Iris Flower	82
4.3	Applying OAR to the Iris Flower Workflow	85
4.3.1	Identifying Relevant Archetypes and Constraints	86
4.3.2	Formulating the Solution Specification	86
4.4	Implementing the Solution Specification	87
4.5	Summary and Conclusion	88
5	CONTEXTUALIZING COURSE DESIGN	91
5.1	Learning, Teaching and Course Design	92
5.1.1	Learning Outcomes	94
5.1.2	Translating Learning Outcomes to Course Design	95
5.2	Contextualizing Course Design	96
5.3	Translating Learning Outcomes for COMP8120	97
5.3.1	Learning Outcomes for COMP8120	97
5.3.2	Analyzing Context for LO-1 and LO-2	98
5.3.3	Analyzing Context for LO-3	98
5.3.4	Analyzing Context for LO-4	99
5.3.5	Analyzing Context for LO-5	99
5.4	Solution Specification for LO-5	99
5.4.1	Initializing External Shelves	99
5.4.2	Identifying Relevant Archetypes and Constraints	100
5.4.3	Solution Shelf for LO-5	101
5.4.4	Implementing the Solution Specification	102
5.5	Summary and Conclusion	102
6	MANAGING LARGE AND COMPLEX SYSTEMS	103
6.1	Large and Complex Systems	104
6.2	Some Characteristics of Large Systems	105
6.3	How complexity builds and escalates in large systems	106
6.4	Applying OAR to Complex Systems	107
6.5	The Ubuntu Platform as a Complex System-of-Systems	109
6.6	Capturing intellectual concerns for the Ubuntu ecosystem	109
6.6.1	Initializing External Shelves	110

6.7	Identifying Relevant Archetypes and Constraints	111
6.8	Solution Specification for Selecting the Default Music App	111
6.9	Utilizing a Solution Specification	113
6.10	Summary and Conclusion	117
IV	Conclusion	119
7	SOMA: OAR TOOL PROTOTYPE	121
7.1	Soma: A Tool for Simple Omnispersive Analysis and Reasoning	122
7.1.1	Initialization	122
7.1.2	Building the Problem–domain Shelf	123
7.1.3	Contextualization	124
7.1.4	Practical Considerations in Soma	124
7.2	Product Vision and Goal	126
7.3	Architecture Vision and Sprint Planning	126
7.3.1	Architecture Vision	127
7.4	Soma Development	128
7.4.1	Soma Sprint 1	128
7.4.2	Soma Sprint 2	135
7.5	Summary and Conclusion	137
8	SUMMARY AND CONCLUSIONS	141
8.1	Summary of Contribution	142
8.1.1	An Enhanced Definition of Scientific Workflow	142
8.1.2	Omnispersive Analysis and Reasoning	142
8.2	Limitations of Contribution	145
8.3	Related Work	147
8.4	Viewing Enterprise Architecture through OAR	148
8.4.1	Enterprise Architecture and Architecture Frameworks	149
8.4.2	Applying OAR — Architecture Views as Workflows	152
8.5	Directions for Future Work	155
8.5.1	Managing Fractional Values for Firmness and Influence in Recipe Context	155
8.5.2	Tool Support	155
8.5.3	Moodle OAR Plugin	156
8.5.4	Application to Complex Systems	156
8.5.5	OAR as Science of Workflows	156
	BIBLIOGRAPHY	157

LIST OF FIGURES

Figure 1.1	Activity diagram depicting the structure and flow of ideas and results throughout the thesis.	6
Figure 2.1	Components and interfaces in the workflow reference model (after Hollingsworth [1995]). Copyright ©1995 The Workflow Management Coalition. Adapted with permission.	16
Figure 2.2	Delineating high-level concerns and low-level details in an experiment.	25
Figure 2.3	Illustrating context.	28
Figure 3.1	An adaptation of Boyd's OODA loop for managing complex problem situations (after [Flint, 2008]). Copyright ©2008 Flint, S. Adapted with permission.	37
Figure 3.2	The Domain of Science Model (after [Warfield, 1994]). Copyright ©1986 International Society for the Systems Sciences. Adapted with permission.	43
Figure 3.3	A definition of the OAR framework.	52
Figure 3.4	Composite nature of Unit knowledge entity (uke).	55
Figure 3.5	Hierarchy of concerns	56
Figure 3.6	Uke context as a function of Firmness and Influence.	57
Figure 3.7	Relation between concern, recipe and uke.	62
Figure 3.8	Shelves and recipes.	64
Figure 3.9	Managing concerns in the OAR framework.	65
Figure 3.10	Visualization of shelf bootstrap.	67
Figure 3.11	Representing context by profiles of the attributes Influence and Firmness.	68
Figure 3.12	The process of context refinement.	69
Figure 4.1	Workflow for folding the Iris flower starting with a Frog Base.	83
Figure 4.2	Illustrating the implementation-level focus of current scientific workflow practice.	84
Figure 4.3	External shelves and prototypes.	85
Figure 4.4	Selected prototypes in the Problem-domain shelf for the Iris Flower specification.	86
Figure 4.5	Origami Iris Flower specification.	87

- Figure 4.6 Formulating a solution specification for implementing the Iris Flower solution shelf of [Figure 4.5](#). 89
- Figure 5.1 A model of curriculum design. Copyright ©2010 Johns-Boast, L. Adapted with permission. 93
- Figure 5.2 External shelves and prototypes. 100
- Figure 5.3 Archetype and constraint identification for LO-5. 101
- Figure 5.4 Solution specification for LO-5. 101
- Figure 6.1 Escalation of problem complexity when additional inputs are provided. 106
- Figure 6.2 External shelves and recipes for deciding the default music app. 110
- Figure 6.3 Archetype and constraint identification in the problem domain shelf. 111
- Figure 6.4 Solution shelf for the default music app specification. 113
- Figure 6.5 Local ontology for default apps. 115
- Figure 6.6 The problem-domain shelf for selecting the default note app. 116
- Figure 6.7 In this recipe, the Ubuntu 12.04 LTS distribution does not ship with a default note app. 116
- Figure 6.8 In this recipe, the Ubuntu 12.04 LTS distribution can ship with GNote as the default note app even though it lacks critical synchronization features. 117
- Figure 7.1 The initial state of Soma with several external shelves and an empty problem-domain shelf canvas. 123
- Figure 7.2 The newly initialized PDS is populated 124
- Figure 7.3 Possible ambiguity due to visual simplification of constraint representation in a solution shelf. 125
- Figure 7.4 The requirements pyramid for the Soma OAR tool prototype. 127
- Figure 7.5 Sliced view of the Soma architecture cluster. 127
- Figure 7.6 Data architecture model of Soma. Storage of OAR data is independent of its representation in the Soma Logical Layer. The Soma Visual Layer can be customized to generate different outputs. 128
- Figure 7.7 Soma data ring during Sprint 1. 129
- Figure 7.8 Soma data model. 130
- Figure 7.9 Opening screen of Soma. 131
- Figure 7.10 Creating a new recipe. 132

Figure 7.11	Filling in the details for a new recipe. The recipe UUID is auto-generated.	132
Figure 7.12	A recipe for the Flat technique.	133
Figure 7.13	A recipe for Valley fold.	133
Figure 7.14	Entering the details of a new specification.	134
Figure 7.15	Recipes in the Problem-domain Shelf view.	134
Figure 7.16	Assigning recipe state.	135
Figure 7.17	Constraints for the origami iris flower specification. The <i>Frog Base</i> , <i>Mountain Fold</i> and <i>Flap</i> recipes are not shown in this figure.	136
Figure 7.18	Selecting recipes for context refinement.	136
Figure 7.19	Assigning recipe influence.	137
Figure 7.20	Saving the solution specification.	138
Figure 8.1	OAR process for generating architecture views.	154

LIST OF TABLES

Table 3.1	Ideas in the OAR framework.	72
Table 3.2	Some OAR translation engines.	73
Table 7.1	Soma release and sprint planning.	129

Part I

Introduction

1

OVERVIEW

The White Rabbit put on his spectacles. 'Where shall I begin, please your Majesty?' he asked.
'Begin at the beginning,' the King said gravely, 'and go on till you come to the end: then stop.'

Carroll [2008]

1.1	Initial Motivation and Research Aim	4
1.2	Research Design	4
1.3	Thesis Scope	5
1.4	Thesis Structure	5
1.4.1	Part I: Introduction	5
1.4.2	Part II: Omnispective Analysis and Reasoning	7
1.4.3	Part III: Proof of Concept	7
1.4.4	Part IV: Conclusion	7
1.5	Main Contributions	7

This thesis is submitted for the degree of Doctor of Philosophy of the Australian National University. Exploratory research leading to the conception, development and demonstration of Omnispective Analysis and Reasoning (OAR), an epistemic framework to enhance the management of intellectual concerns in scientific workflows is described in this thesis. The application of the framework to manage ideas and concepts in generic workflows and large and complex systems management is also discussed.

An introduction to the research and information about the structure and organization of the remainder of the thesis is given in [Chapter 1](#). This chapter also provides a high-level overview of the research and the main contributions reported in the thesis.

1.1 INITIAL MOTIVATION AND RESEARCH AIM

The research leading to the development of the OAR framework commenced with an initial aim to study scientific workflow management in order to understand and improve upon the handling of intellectual concerns (ideas, concepts and hypotheses) in scientific processes and research.

The results of preliminary research, which are presented in [Chapter 2](#), lead to the conjecture that the handling of intellectual concerns in scientific workflow management can be improved by adding a layer of abstraction to shift focus from low-level implementation details. This can be further improved by adding context support to scientific workflow components and processes and providing a suitable level of abstraction to capture and organize the underlying concepts, theories, ideas and rules in the problem domain being explored and map them to the workflow specification and execution semantics. An amplification of these concepts is used to formulate and develop Omnispective Analysis and Reasoning (OAR), an epistemic framework to effectively manage the associated intellectual effort in scientific workflows.

Accordingly, the research aim of this thesis can be stated as: *Formulate, develop and demonstrate an epistemic framework for managing intellectual concerns in scientific workflows.*

1.2 RESEARCH DESIGN

As stated above, a key motivation for undertaking the research documented in this thesis is a need to improve the management of intellectual concerns in scientific workflows.

A two-phase approach to Omnispective Analysis and Reasoning research is utilized using the *Engineering Method of Research*, which is outlined by Flint [2009]. The first phase comprises the theoretical work and proof of concept presented in this thesis. The second phase constitutes future research in the OAR framework. The research has been conducted by observing the use of current approaches to intellectual concern management in scientific workflows ([Chapter 2](#)), and then conceptualizing, formulating and demonstrating the OAR framework for managing intellectual concerns (chapters 3 to 6).

1.3 THESIS SCOPE

Aside from meeting the research aim stated above, the scope of work also includes the illustration of the applicability of OAR. This is presented in chapters 4 to 6. However, extensive evaluation and quantification of the framework is beyond the scope of the thesis.

An approach to develop tool support for OAR is suggested and a prototype implementation is presented in Chapter 7. The full analysis, design and implementation of a complete end-to-end OAR toolset is outside the scope of the work discussed in this thesis.

1.4 THESIS STRUCTURE

Following the approach presented by Flint [2006], the structure of this thesis is depicted in Figure 1.1 using a Unified Modeling Language (UML) activity diagram [Mellor and Balcer, 2002]. Each chapter of the thesis is represented by an activity (rounded box). The flow of ideas and research results is represented by directed arrows between the chapters. Key contributions of this research are depicted by objects in the diagram (white square boxes).

As indicated by the vertical partitions in Figure 1.1, this thesis is organized in four parts.

A detailed overview of the research can be obtained by reading chapters 2, 3 and 8. The overall conclusion is presented in Chapter 8. The OAR framework itself is presented in Chapter 3.

1.4.1 Part I: Introduction

As indicated in Figure 1.1, Part I comprises two chapters. The motivation and aim for the research are presented in Chapter 1. The background for OAR is discussed in Chapter 2. This includes a historical overview of scientific workflow management to identify approaches to address the problem of managing intellectual concerns. As also indicated in Figure 1.1, this study resulted in the formulation and development of Omniscient Analysis and Reasoning.

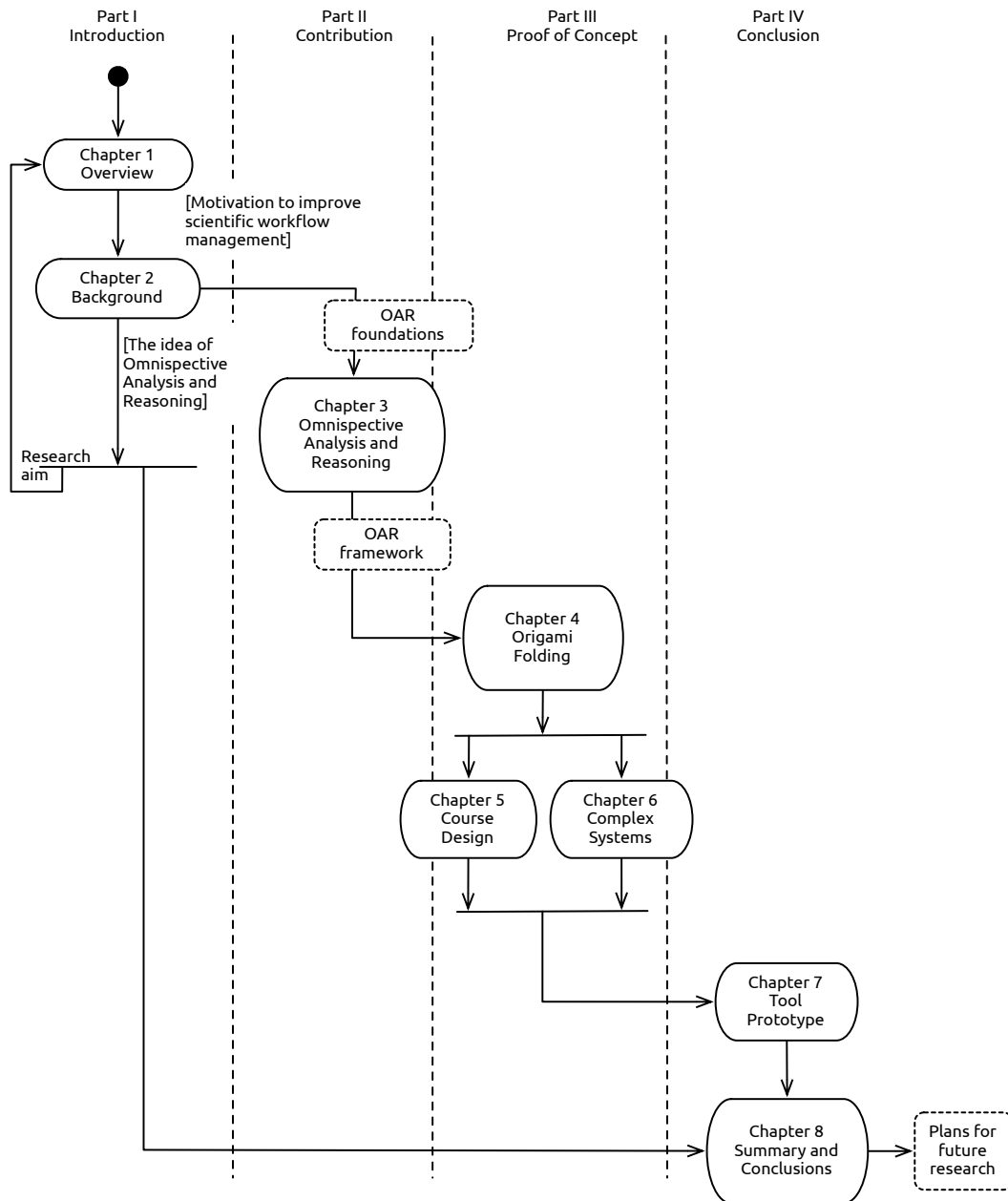


Figure 1.1: Activity diagram depicting the structure and flow of ideas and results throughout the thesis.

1.4.2 [Part II](#): Omniscpective Analysis and Reasoning

[Part II](#) of this thesis comprises one chapter. The OAR framework and its foundations are described in [Chapter 3](#). The OAR framework has been designed to bridge the cognitive chasm between the underlying theory, rationale and concepts and the design and execution of workflows in scientific processes.

1.4.3 [Part III](#): Proof of Concept

The application of the OAR framework is illustrated by three examples in [Part III](#). The focus of these applications is to clearly demonstrate the working and potential of the OAR framework across diverse disciplines. The application of OAR to an origami iris flower workflow is presented in [Chapter 4](#). The second illustration presents the use of OAR in contextual course design ([Chapter 5](#)). The applicability and potential of OAR in analyzing, understanding and managing complex systems is demonstrated in [Chapter 6](#).

1.4.4 [Part IV](#): Conclusion

The development of a tool prototype for OAR is presented in [Chapter 7](#). Summary, discussion and conclusions are given in [Chapter 8](#). Directions for future work and research are also presented.

1.5 MAIN CONTRIBUTIONS

The research reported in this thesis has resulted in the following contributions to existing knowledge required to understand and manage intellectual concerns in scientific workflow management.

1. *Omniscpective Analysis and Reasoning*: OAR is a novel epistemic framework to identify, capture and reuse intellectual concerns (ideas, concepts and hypotheses) in scientific workflows ([Chapter 3](#)).
2. *Definition of scientific workflow*: An extended and enhanced definition of scientific workflow beyond the currently understood scope of “computer flowcharting” ([Section 3.9](#)).

3. *Hierarchy of Conceptualization*: A hierarchy for the conceptualization of an entity at the concept, model and implementation levels is presented ([Section 3.5](#)).
4. *Idea of localized ontologies*: The idea of localized ontologies, applicable to a particular instance of a problem situation, is introduced and realized through shelf management in OAR. Localized ontologies assist the process of arriving at shared semantic understanding by reducing the scale and complexity of the ontological mapping ([Section 3.11.4](#) and [Chapter 6](#)).

2 | BACKGROUND

Listen to others as you would have others listen to you.

Booth, Colomb, and Williams [2008]

2.1	Workflows	10
2.1.1	Scientific Database Systems	10
2.1.2	Unstructured Activities	11
2.1.3	Dynamic Process Composition	13
2.1.4	Distributed and Decentralized Processes	13
2.1.5	The Workflow Reference Model	15
2.1.6	Participatory Analysis	17
2.1.7	Emergence of Scientific Workflows	18
2.1.8	Scientific Workflows	20
2.2	Issues in Scientific Workflow Management	23
2.2.1	Focus on Low-level Detail	23
2.2.2	Limited Context Support	26
2.2.3	Inadequate Management of Intellectual Concerns	28
2.3	Impact of the Above Issues	29
2.3.1	Observation 1	30
2.3.2	Observation 2	30
2.3.3	Observation 3	30
2.4	Conjecture	31
2.5	Summary and Conclusion	32

This chapter commences with a discussion of workflows and workflow management systems. The evolution of scientific workflow systems as distinct from business workflow systems is traced in [Section 2.1](#). Some of the issues that need to be addressed in order to improve the management of intellectual concerns in scientific workflows are identified in [Section 2.2](#). The impact of these issues is considered in [Section 2.3](#). Based on these observations, the conjecture leading to the formulation and development of Omnispersive Analysis and Reasoning is presented in [Section 2.4](#).

2.1 WORKFLOWS

Scientific workflows are generally considered as a formal representation of the processes undertaken by scientists or researchers in the investigation and exploration of a problem situation. Over the years, the terms scientific workflow and scientific workflow management have taken on different meanings in different contexts and disciplines, being mostly used as generic terms to describe any dynamic series of structured activities and computations that are routinely encountered in scientific problem-solving [Singh and Vouk, 1996; Wainer et al., 1997].

The Workflow Management Coalition (WFMC) defines a workflow and workflow management system as [Hollingsworth, 1995]:

Workflow: The computerized facilitation or automation of a business process, in whole or part.

Workflow Management System: A system that completely defines, manages and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic.

The concept of workflows and workflow management originated in the activities to manage, organize and optimize business and office processes [Ludäscher et al., 2009; Medina-Mora et al., 1992]. Business workflow management systems were used to handle the challenges of automation of increasingly complex processes such as inventory control, personnel and payroll management, purchase order processing and financial management in business and office environments. Various workflow management systems such as Bonita, JBoss and YAWL are geared towards specifying, enacting and managing business processes [Garces et al., 2009].

The use of workflow management systems in the scientific domain led to the development of customized workflow management paradigms specific to exploratory research, analysis, scientific data management and execution of complex and computationally intensive experimental processes which require keeping track of data, activities and resources which are geographically distributed and structurally heterogeneous.

2.1.1 Scientific Database Systems

Shoshani, Olken, and Wong [1984] proposed a type classification for scientific data, and identified the following characteristics which distinguish scientific data management from business data processes:

1. Scientific data is typically multidimensional with a high occurrence of sparse data sets.
2. Most searches of scientific data routinely exhibit a high degree of locality.
3. Though most scientific databases are usually quite large, they are also quite modular in structure. This makes it possible to design a structural decomposition which is better suited to scale efficiently in terms of organization and storage.
4. A large part of scientific data is made up of results generated by associated scientific instrumentation. Thus it is desirable to store information about the pertinent instrumentation along with the data generated.
5. Summary analysis of scientific datasets is common enough to warrant specialized support to optimize the process. This analysis encompasses a varied set of data types exhibiting a high degree of mix-in, which is quite atypical of traditional mainstream database management where data sets tend to be more uniform.

Based on these observations, Shoshani, Olken, and Wong concluded that scientific data management requires features which are not generally available in business databases. They proposed that it might be desirable to have a class of specialized database management systems which are tailored specifically to scientific data and its associated instrumentation.

These developments, coupled with the particularly challenging requirements posed by the management of scientific data [Shoshani, Olken, and Wong, 1984], and the disparities between exploratory scientific research and technological processes in industrial production [Ioannidis et al., 1997] resulted in additional requirements leading to the branching of scientific workflow development from general workflow management systems for business and office use.

2.1.2 Unstructured Activities

Subsequent advances in computational research made it feasible to consider the use of automation to capture process logic and coordinate workflow control. Medina-Mora et al. [1992] describe and present a procedure for office work (business workflows) using a number of interconnected loops, where each loop represents an elementary function as a transaction between a customer and a performer. This approach, while generic enough in theory to represent workflows in scientific processes, was primarily focused on realizing efficiency gains in office procedure automation to enhance customer satisfaction.

The recognition of these specific needs to scientific data management listed above vis-a-vis data and process automation saw the advent of *Experiment Management Systems (EMS)* which aimed to enable the generation and management of experimental scientific data. The MOOSE Experiment Management System [Wiener and Ioannidis, 1993] was developed to support experimental data management via a specialized query language. Experimental objects in scientific processes and their associated data were stored in customized Object-Oriented Database Management Systems and manipulated using a specialized declarative query language. This approach sought to address the special concerns of data and process locality, periodicity in temporal data sets and the maintenance of interconnected data sets and their generating processes.

Increasing mainstream use of workflow management processes in business organizations led to further research in enhancing workflow management practices to accommodate exception handling. Workflow exceptions were now handled at two levels of management — the workflow description or modeling level, and the workflow execution level. A mechanism for handling exceptional data and processes in workflow management was proposed by Barthelmeß and Wainer [1995]. This approach presented one of the early abstractions for managing workflow processes at two distinct levels of granularity. Issues dealing with workflow user-interface concerns, data organization and process schematics were grouped together in the *workflow modeling level*. Workflow implementation was handled in the *execution level* with support for exception handling in workflow enactment.

Adding support for exception handling was the first step for building support for unstructured activities in workflow management processes. However, the flexibility offered to customize the workflow process as per suitability still remained limited until the introduction of *contextual information agents* [Blumenthal and Nutt, 1995]. Though rather limited in scope and form at this stage, the addition of context information to unstructured activities enabled their inclusion in the hitherto highly structured and rigid workflow process specifications. The Bramble context information agent [Blumenthal and Nutt, 1995] added support for the following context variables as a tuple to better specify the scope of the workflow process:

1. A description of the context environment.
2. A statement of goals of the activity within the purview of the context description.
3. A set of constraints which governed the context criteria; and

4. A specification of the user role participating in the workflow process.

This rather limited view of context as a scope delimiter remained prevalent even with further developments in workflow execution methodologies.

2.1.3 Dynamic Process Composition

The diversification of scientific workflow processes from business workflows continued with further work to address the dynamic nature of scientific research. Bogia and Kaplan [1995] presented a model for handling dynamic changes to workflows. This approach handled dynamic exceptions locally within the entire workflow with support for composing the individual *sub-workflows* together. This enabled better handling of modifications that were not shared in the entire workflow, and were local to a particular context of execution. This *overlay* metaphor to workflows added support for workflow inheritance, while delegating the decision-making in the composition process to a human agent, still providing the ability to incorporate automation features from decision support systems.

A major consequence of dynamic workflow composition was the increasing need to address issues due to process and data parallelism. An approach to manage parallelism in workflow processes was presented by Ellis, Keddara, and Rozenberg [1995]. They classified dynamic changes as structural and component changes and presented a petri-net based approach for tackling dynamic structural changes. Dynamic component changes affect workflow components such as roles, repositories and resources, while dynamic structural changes are limited to the changes *within the structure of* procedures. A chief use of the petri-net approach for managing structural dynamism is in the modeling component, without undue emphasis on the execution and implementation of the workflow. Further developments in handling dynamic evolution in workflows have been reported [Casati et al., 1998; van der Aalst and Basten, 2002]. However, none of these approaches make any mention of features for ensuring correctness or validity of the modeled structural changes.

2.1.4 Distributed and Decentralized Processes

The growing dichotomy of business and scientific workflows was further concretized by the development of database management systems specifically customized for scientific processes and data.

Medeiros, Vossen, and Weske [1995] outline an architectural framework to support scientific applications in which huge amounts of data have to be managed

efficiently and in an application-specific way. The framework is constructed with distinct layers of abstraction and functionality and can be built atop arbitrary existing types of databases. The operation of the framework crucially centers around the paradigm of database management for scientific applications.

Compared to previous customization of database systems for scientific applications, Workflow-based Architecture to support Scientific Applications (WASA) [Medeiros, Vossen, and Weske, 1995; Weske, Vossen, and Medeiros, 1996] approaches scientific workflow management with a layered paradigm. Similar to data warehouse management systems [Han, Kamber, and Gray, 2000], workflow management is handled over four layers:

1. A *user interface layer* to handle communication between users and the system.
2. An *internal tool layer* to coordinate experiment specification, documentation and execution.
3. An *enhanced database functionality layer* to present a unified interaction layer to the internal tool layer, and
4. A *database layer* to interface with the underlying database systems used in the application.

Though WASA exhibits a hierarchical abstraction of workflow management concerns, the end user is not insulated from the low level architectural details even at the user interface layer.

In contrast to earlier attempts at workflow abstraction, WASA views all workflow activities as steps and tasks in an experiment, and aims to support their specification, control and execution by delineating workflow execution semantics from architectural concerns. A primary application for this pipeline architecture was demonstrated in model sequencing in DNA fragment assembly [Meidanis, Vossen, and Weske, 1996].

A parallel development in workflow evolution was the emergence of decentralized workflow management. As opposed to a central task director, this approach was based on the concept of a workflow component mediator which was built upon existing commercial middleware platforms utilizing the Object Management Group (OMG) Common Object Request Broker Architecture (CORBA). Schill and Mittasch [1996] implemented a strongly structured view of workflow tasks on top of the CORBA Internet Inter-ORB Protocol (IIOP) to handle communication between remotely located business workflow objects. Each workflow object which is located on a participating node exposes an invocation interface to the workflow coordinator.

A paradigm shift from traditional Experiment Management Systems (EMS) and Engineering Data Management Systems (EDMS) to comprehensive decentralized scientific workflow management was demonstrated by Baker, McClatchey, and Le Goff [1997]. The CRISTAL (Concurrent Repositories and Information System for Tracking Assembly and production Lifecycles) monitoring and production system [Baker, McClatchey, and Le Goff, 1997] incorporated the management of workflow instantiation and enactment processes over an object-oriented database layer. Their system considered each evolving and changing workflow specification as a one-off production workflow leading to a refined deployment workflow in the end. This enabled them to track the workflow changes and their correlated constructs in order to execute and evaluate partially developed workflow specifications. The use of meta-objects and indirection to support dynamic workflow changes was considered, but not implemented owing to the added complexity and impact on efficiency in the CRISTAL system.

Enhancements in computing power of commodity desktop computers further facilitated the development of scientific workflows. A synthesis of scientific database management and scientific workflow concepts was put forward by Ioannidis et al. [1997]. The ZOO Desktop Experiment Management Environment was developed as an experimental lifecycle management system for individual researchers to specify, execute and manage various phases of the research lifecycle, with particular customizations in the domain of soil science. The emergence of individual experiment management systems for scientists led to a further interest in research pertaining to the sharing, comparison and collation of experimental data and procedures in addition to managing the processing of scientific workflows. Further developments have incorporated dynamic coupling to manage distributed workflow processes [Akarsu et al., 1998; McClatchey et al., 1997; Reuß, Vossen, and Weske, 1997].

2.1.5 The Workflow Reference Model

With a view to standardizing the growing diversity of workflow management practices and approaches, the WFMC was established in 1993 as a non-profit international body for the development and promotion of standards for software terminology, interoperability and connectivity between workflow products. A chief contribution of the WFMC was the proposal for a workflow reference model for business workflow processes [Hollingsworth, 1995].

The workflow reference model describes an architecture for the major components and interfaces of workflow systems.

All workflow systems contain a number of generic components which interact in a variety of ways. The model shown in [Figure 2.1](#) identifies the major components and interfaces of the workflow reference model:

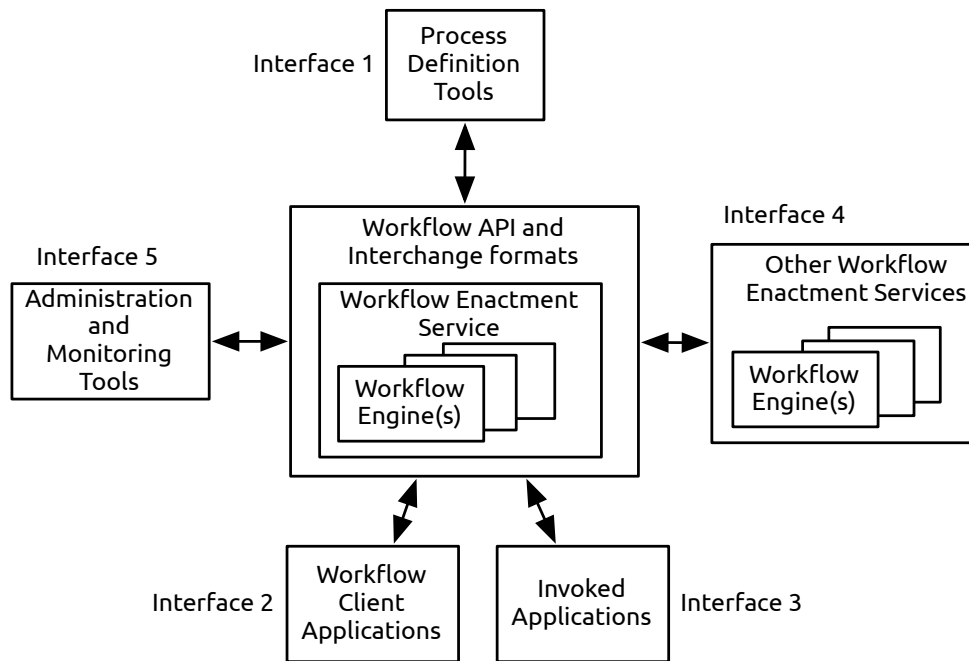


Figure 2.1: Components and interfaces in the workflow reference model (after Hollingsworth [1995]). Copyright ©1995 The Workflow Management Coalition. Adapted with permission.

1. Process Definition Tools are used to analyze, model and describe business processes.
2. Workflow Enactment Service is the run-time environment where workflow processes are executed. This may involve multiple workflow engines. This service is responsible for reading process definitions and creating and managing process instances.
3. Workflow Client Applications are the software entities which present items to the end user, invoke application tools which support the task and the data related to it, and allow the user to take actions before passing control back to the workflow enactment service.
4. Administration and Monitoring Tools can be used to track process status for control, management and analysis purposes.

The reference model defines five interfaces to the workflow enactment service:

1. Process definition import/export interface.
2. Interaction with workflow client applications and software for presentation of worklists.
3. Tools and external application invocation.
4. Interoperability between several workflow management systems.
5. Interaction with administration and monitoring tools.

The workflow reference model defines a common interchange format, the Workflow Process Definition Language (WPDL), which supports the transfer of workflow process definitions between separate products.

A set of extensibility mechanisms to support vendor specific requirements is proposed in the WPDL definition. This is based on the definition of a Workflow Meta-Model, a limited number of entities that describe a workflow process definition. The meta-model identifies a basic set of entities and attributes for the exchange of processes: Process Definition, Process Activity, Participant Definition, Transition Information, Application Definition, and Process Relevant Data. These entities contain attributes which support a common description mechanism for processes. Further entities and attributes may be added to the model to create future conformance models.

The WFMC reference model also includes one representative business case that can be used to verify the feasibility of the implementation of the standard, as well as to constitute a preliminary test of a conformance assessment procedure.

Lin et al. [2009] adapted the workflow reference architecture to scientific workflows, by suggesting six interfaces instead of five — assigning separate interfaces for data management and program management. They supposed that this separate interface would be able “to support scientific result reproducibility and to facilitate and speed-up data analysis.”

2.1.6 Participatory Analysis

The review of the evolution of workflow management practices thus far identifies the following emerging research themes [Shi et al., 1998]:

1. Flexibility in workflow modeling and process specification in order to provide support for temporal reordering of repetitive processes.
2. The use of object-oriented workflow management systems to map distributed object repositories to workflow coordinators.

3. Dealing with issues of workflow synchronization and coordination among distributed workflow participants.
4. The development and customization of web-based interfaces for existing workflow systems.

Work on providing support for adaptive exploratory workflows [Adelsberger, Körner, and Pawlowski, 1999; Han, Sheth, and Bussler, 1998] sought to optimize automated workflows by interfacing them with computer assisted learning environments. The requirement of pre-structured and pre-modeled workflows, a remnant of the business domain origins of workflow practices, was rapidly discarded in favor of dynamic approaches for managing scientific workflow processes. While Adelsberger, Körner, and Pawlowski [1999] focused primarily on connecting workflow processes to computer-assisted learning systems in order to optimize business processes using rule-based inference sets, Chin et al. [2000] investigated the application of interactive problem solving environments to scientific workflow systems. The increasing complexity of interacting workflows [Bussler, 1999] gave rise to approaches to make distributed workflow processes more tractable and reusable [van der Aalst et al., 2000, 2001]. These participatory and interactive processes are characteristic of scientific research.

Additionally there is a high degree of uncertainty coupled with added requirements of openness, interoperability, sharing and connectivity in scientific activities [Adelsberger, Körner, and Pawlowski, 1999] as compared to the pre-modeled, highly structured and heavily goal-oriented proprietary nature of business and commercial processes [Bussler, 1999]. Business workflow systems therefore lack, in general, the requisite degree of customizability and flexibility to be easily adaptable for scientific process management without extensive reengineering and changes.

Thus, the development of scientific workflows and their management was pursued independent of and concurrently with business workflows laying firm foundations for the specific field of scientific workflow management systems.

2.1.7 Emergence of Scientific Workflows

Following on these developments, a formalization of the term *scientific workflow* began to emerge [Singh and Vouk, 1996]. This approach sought to bring together various customizations of workflow practices from traditional business management domains under the specific grouping of scientific workflow management. The following main features of scientific workflows were identified:

1. Scientific workflows are typically used in research and exploratory analysis domains.
2. These involve activities revolving around computation-intensive research practices.
3. As a result, they require robust support for extensive parallelization of tasks and computations.
4. Scientific workflows are a custom specialization of concepts, processes and methods utilized in traditional business workflows.
5. There is a high incidence of unstructured activities in scientific workflows in the initial exploratory phases of research.
6. Composition of scientific workflows is highly dynamic, often requiring changes while the workflow is being specified and executed.
7. A common approach in the development of production workflows is the refinement and optimization of exploratory scientific workflows. Initial problem analysis is undertaken using a highly dynamic scientific workflow formulation. As the workflow matures and develops into a stable process specification, it can be standardized and deployed as a production workflow in a specific business domain.

Based on these characteristics, the term *scientific workflows* was put forward [Singh and Vouk, 1996]:

We use the term *scientific workflows* as a blanket term to describe series of structured activities and computations that arise in scientific problem-solving.

Wainer et al. [1997] further identify the following distinguishing characteristics between business workflows and the newly emerging paradigm of scientific workflows:

1. Both business and scientific workflow management systems are primarily used to control and organize the sequence of processes in a business or scientific procedure. Business processes are primarily goal driven, whereas the main focus of scientific workflows is data management.
2. As a consequence, business workflows are heavily structured and modeled prior to enactment, with little modification while the workflow is executed. On the other hand, scientific workflows require a greater amount of flexibility in collating and analyzing data from heterogeneous sources.

3. Business workflows are usually executed with a stated *endpoint* that signals the termination of the sequence of steps. The exploratory nature of scientific workflows means that a given data step may be either entirely abandoned or restructured while the workflow is being executed.

With these distinctions in mind, Wainer et al. [1997] coined the term *ad hoc* flow for scientific workflows that need to be adaptive enough to either alter, repeat or reroute intermediate process steps while the workflow is executing. An additional characteristic of ad hoc flows is the shift in emphasis away from pre-modeled workflows to workflow processes that are scripted in response to exceptional situations, as suggested earlier by Barthelmeß and Wainer [1995].

2.1.8 Scientific Workflows

Scientific workflows are being used extensively to support the management of experiments and computational research in various scientific domains in order to support large-scale data management, control and coordination of computationally intensive processes, and the allocation, sharing and organization of resources in complex scientific environments [Barker and Hemert, 2008; Ludäscher, Bowers, and McPhillips, 2009].

Scientific workflow management systems in vogue are primarily designed to fulfill the following two requirements:

1. Enable the scientist to focus on issues of scientific research, and to delegate to the scientific workflow management system all or most of the chores associated with managing the details of the software systems and instrumentation used in the research.
2. Automate repetitive and computationally intensive tasks associated with data gathering and analysis, which are highly error-prone if carried out manually owing to the complexity of the systems involved. This is necessary in order to efficiently utilize shared resources and systems by optimizing the details of process and task scheduling.

Pursuing this path, various scientific workflow management systems like Kepler [Altintas et al., 2004], Taverna [Oinn et al., 2004b], VisTrails [Callahan et al., 2006a] and Triana [Churches et al., 2006] have been built to facilitate research activities in different domains of scientific research.

All of these scientific workflow management systems adhere to the following definition of a scientific workflow [Ludäscher, Bowers, and McPhillips, 2009]:

A scientific workflow is the description of a process for accomplishing a scientific objective, usually expressed in terms of *tasks* and their *dependencies*.

It may be noted that the term scientific workflow has been established by ad hoc usage. References like “typical scientific workflows” and circular definitions are often encountered in literature. Thus, it is necessary to provide an improved definition of scientific workflows. This issue is addressed in [Chapter 3](#).

Current scientific workflow systems provide features to design and orchestrate experimental and computational steps in scientific data collection, organization and analysis [Baker, McClatchey, and Le Goff, 1997; Curcin and Ghanem, 2008; McPhillips et al., 2009; Weske, Vossen, and Medeiros, 1996]. The user can specify the steps to run for each experimental process, which data to draw from and connect to, and publish data and results in shared repositories and databases. This allows better exception handling in the research and supports automation of repetitive and computationally intensive tasks.

The key features of a few popular scientific workflow management systems are presented below.

Kepler

The Kepler scientific workflow management system [Altintas et al., 2004; Ludäscher et al., 2006] uses a data-centric approach [Shields, 2007] to scientific workflow management. Data is input from databases of interest and the workflow is designed using workflow actors to link and execute process stubs on a workflow canvas. A workflow director is defined to coordinate workflow design and execution.

Kepler provides the following features to assist the specification and execution of the scientific workflow:

1. Kepler uses an actor driven modeler to support rapid prototyping of workflows. Participating components are represented as actors in the workflow specification which is then compiled to execute and run the workflow.
2. The support of shared components enables interaction with distributed workflow services. This allows grid-enabled workflow systems such as Pegasus [Deelman et al., 2004] and Webflow [Akarsu et al., 1998] to coordinate and share data and results with the Kepler workflow.
3. A scientific workflow specified in Kepler can accept data input from a variety of supported data interchange formats using XSLT-based data transformation actors.

4. Provenance information about workflow actors, data inputs and process control is provided. Invocation dependency graphs help manage and establish process provenance. Data provenance is supported by flow graphs and data lineage graphs.

The Kepler workflow system has been used in research and analysis in various fields such as astrophysics, biology, chemistry and ecology [Altintas et al., 2004].

VisTrails

VisTrails [Callahan et al., 2006a,b] is a visualization based scientific data and workflow management system. The main focus is on presenting an accessible visualization of the experimental data and its provenance in order to support the scientist in making an informed selection of relevant datasets.

In addition to support for workflow modeling and management, VisTrails provides the following features to facilitate the selection and management of large data sets:

1. VisTrails provides support for managing the history of a workflow specification. Versioning information for dataflows presents a clear provenance timeline of the workflow evolution.
2. Support for caching and replaying workflows and dataflows makes it easy to optimize dataflows based on computed results and process outputs without repeating the entire workflow from the beginning. This allows the user to inspect the workflow at different stages of evolution and trace any metadata describing the nature of workflow progression.
3. Inbuilt support for visualization of provenance trails helps capture, annotate and manage rich metadata describing the workflow specifications. The detailed provenance trail allows workflow states to be restored to previously stored checkpoints corresponding to different stages of computations.

The VisTrails scientific workflow management system has been used in visualizing and managing provenance trails in medical imaging, simulation management and uncertainty modeling.

Taverna

Taverna is a specialized scientific workflow management system developed for bioinformatics applications [Hull et al., 2006; Oinn et al., 2004b, 2006].

Taverna provides the following features for scientific workflow management:

1. Workflows are specified in the Simple Conceptual Unified Flow Language (Scufl) [Oinn et al., 2004a], a high-level workflow language in which each step in the process chain is represented as one atomic task. This feature was meant to address the absence of a suitable high-level workflow specification language with features for data, process and resource specification at a semantic level in bioinformatics workflows.
2. A workbench is provided for writing and enacting Scufl workflows using a higher level of abstraction to enable users to write workflows without learning the low level intricacies of Scufl.
3. Communication with interacting services is handled by the *Freefluo* enactor core, a dedicated management layer which provides data sharing and process communication features. Workflows can then be adapted to interface with different services by specifying an enactor interface using the *Freefluo* enactor core.

2.2 ISSUES IN SCIENTIFIC WORKFLOW MANAGEMENT

In scientific research, all the key issues revolve around the intellectual concerns in the problem. Intellectual concerns constitute mainly of research aims and objectives, exploratory domain concepts, scientific models, representation of underlying theories, and process specifications which form the basis of the scientific investigation. The methodologies and processes involved in conducting the activities of experimentation, data collection, data management, computations etc. are well-developed practices in scientific research from even long before the use of computers in these areas. The use of computers enables handling of these issues on a larger scale, yet the steering and guiding importance of the intellectual concerns is what needs to be carefully captured, preserved and used for effective management of scientific research. Explicit and deliberate efforts at incorporating support for the management of intellectual concerns are not noticeable in the current practice of scientific workflow management. The main factors responsible for the inadequate management of intellectual concerns are now examined.

2.2.1 Focus on Low-level Detail

Emphasis and focus in current scientific workflow management practice is more on the logistics of the workflow processes than on ensuring that the appropriate processes are used. This restricted scope of workflow specification, orchestration

and execution often obscures significant cues and feedback which can greatly aid and facilitate in better realizing the primary goals and objectives of the workflow exercise. In addition, even though current approaches to scientific workflow management scale well to solve complex workflow problems typical of a chosen domain of scientific research, none of them mention features to provide additional support for *understanding* and *verifying the rationale* behind workflow orchestration and instrumentation. The lack of this ability to check the *pertinence* and *appropriateness* of process orchestration may often hinder and diminish the relevance and usefulness of the experimental results and analysis.

This could possibly be due to the fact that scientific workflow management systems have been developed as modifications on top of and extending concepts and implementations of business workflow or database management systems. Consequently, scientific workflow management systems also inherit some of the fundamental ad hoc artificiality characteristic of business systems [Meidanis, Vossen, and Weske, 1996].

Though there are occasional references to high-level and low-level representation of “workflow plumbing” mentioned in relation to scientific workflows [Ludäscher et al., 2006, 2009], these represent completely unrelated concepts and the only similarity is one of language. Delineating what constitute high-level concerns and low-level implementation details depends on the nature of the research problem and the granularity at which the goals of the research are set. This delineation is illustrated in [Figure 2.2](#).

When studying a problem at a theoretical level, all other concerns like experimental setup, the design, operational procedures and limitations of the instruments may be considered as low-level details within the theoretical context. On the other hand, while conducting the experiment with the aid of instruments, design principles and limitations of the instruments assume importance as high-level concerns and operational procedures of the instrument (for e.g., switching on, recording the experimental data, preparing tables and graphs as per pre-defined guidelines, etc.) form the low-level implementation details.

Owing to the vastness of even ‘narrow’ fields of specialization, scientific practitioners are often not able to devote attention to all of the high-level concerns involved in the working of the instruments being used. Most times, their working *has to be* accepted only on *faith of certification*, where validation is only implicit. This is, in part, focus on low-level experimental detail without paying attention to the scientific concerns involved in the design and operation of the instrument. Consequently, *valid* results obtained may only be a coincidence. The reliability of the conclusions drawn therefrom may be in question; the researchers may not even be aware of this possible issue.

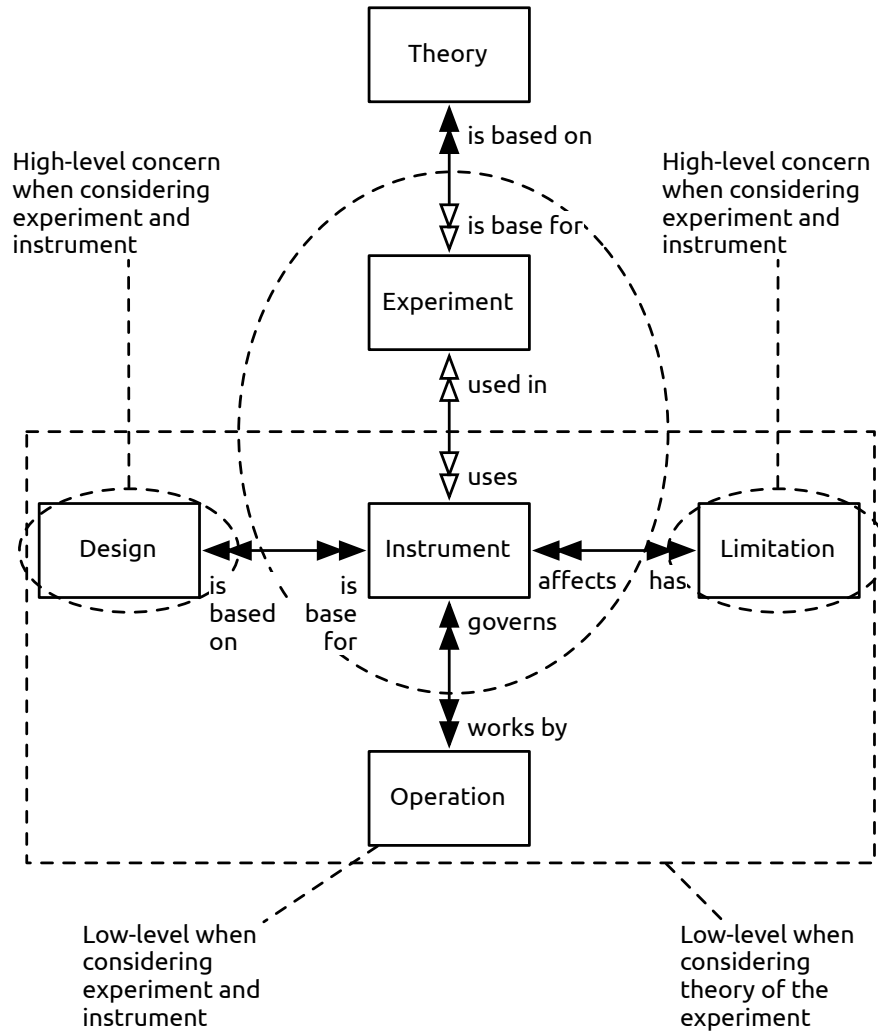


Figure 2.2: Delineating high-level concerns and low-level details in an experiment.

Superficially one may argue that these difficulties occur because of not paying attention to the ‘low-level’ details of instrument calibration. But, in the context of the experiment, the scientific principles involved in the design, calibration and operation of the equipment are not really low-level details of the investigation. This observation is illustrated in the retraction [Marcus, 2011; National Academy of Sciences, 2011] of a research paper owing to the fact that “the experimental data was based on incorrect calibration of the apparatus in viscous solutions and there is no basis for the major conclusions of the study.”

Thus, when the specification of a workflow for scientific systems focuses more on details of data variables, process branching, looping and control, memory allocation and optimization, and other chores connected with the system-level tasks of process and systems control and management, this takes away the focus from the main objective of the scientific activity — forcing users *to lose sight of the forest for the trees* [Chemboli, 2010b]. A similar observation is made by Chin Jr. et al. [2011] who contend that “the basic objects of workflows are too low-level and high-level tools and mechanisms to aid in workflow construction and use are largely unavailable,” and this discourages extensive and widespread development and use of workflows. Pignotti et al. [2011] also observe that existing workflow languages do not provide support for focusing on the scientific constraints and goals (which the authors call “scientist’s intent”) because they are “designed to capture low-level service composition rather than higher-level descriptions of the experimental process.” Additionally, Turuncoglu [2012], in the context of workflows for Earth System Modeling, mentions that the inability to adequately address intellectual concerns “may prevent researchers from focusing on scientific issues, and may make it difficult, or even impossible, to undertake some Earth system science problems.”

2.2.2 Limited Context Support

Although the term *context* has been mentioned in the literature of scientific workflows in the limited sense of ‘conveying a relation within an environment’, no significant attempt seems to have been made to define context and use it as a specific feature of scientific workflow management.

In business and office workflow management, the term context has been used to mean information about the person executing the workflow (who, what, when, where) [Maus, 2001]. In a similar vein, the term context is also used to simply represent the assigned role (workflow creator, designer or user) of the person using the workflow [Ardissono et al., 2007].

The treatment of context in scientific workflows [Chin Jr. et al., 2011; Ngu et al., 2010; Pereira and Travassos, 2010] has been primarily limited to:

1. Information about execution environment (machines used).
2. Information about user (which scientist).
3. Which computation steps (which actors in the workflow).

This view of context is not too different from the limited scope of the term in business workflows. For that matter, it seems no attempts to define and utilize context as a characteristic of software systems have been reported until recently. Alshaikh and Boughton [2009] utilize the definition of context as given by Alexander [1964] and Scharfstein [1991]. They consider an entity as “contextual” or “non-contextual” to a problem in an “implicit” or “explicit” manner.

In this thesis, context is considered as the relationship and influence of a concern on other concerns in the domain of the scientific problem under consideration. It represents the degree of relevance of the concern in the problem domain. A formal definition of context and its representation as used in the OAR framework is presented in [Chapter 3](#).

Here, it should be noted that context as used in OAR is not a material object and does not constitute a defining property of a concern. Context can be realized only in relation to other concerns. It is not absolute to the concern and is an environmental property in this sense. It plays a defining role in the relationship the concern bears with respect to other concerns.

To illustrate the concept of context in the OAR framework, consider the property of a gas, embodied in Boyle’s Law, the relation between the pressure P and the volume V of a gas as:

$$PV = \text{constant}$$

where mass m and temperature T are held constant.

Here, under the context of m and T acting as constraints, P and V are contextually related and result in the property $PV = \text{constant}$ (as illustrated in [Figure 2.3](#)).

Incorporating a definition of context which reflects the relationship between different workflow concerns and their influence on each other in the scientific problem can help establish their role in the workflow formulation and specification. Using context as a formally defined and qualified feature will enable the specification of rich context scenarios to support defining and disseminating the intent and purpose of the workflow specification and execution.

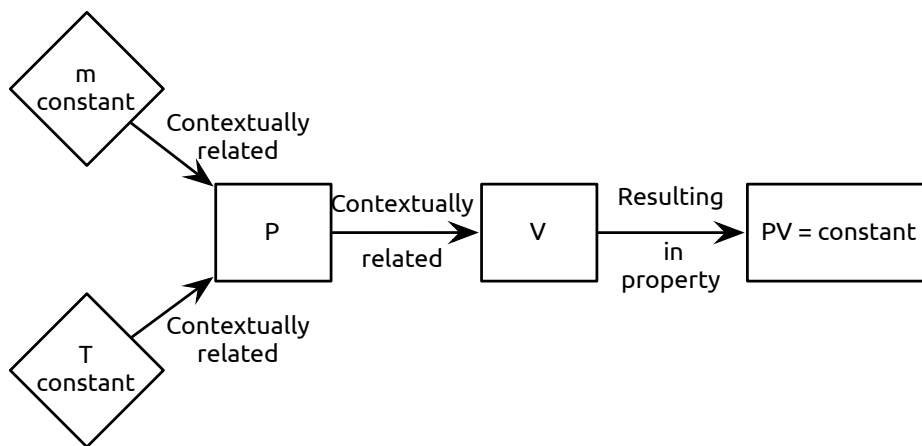


Figure 2.3: Illustrating context.

2.2.3 Inadequate Management of Intellectual Concerns

The capture, organization and dissemination of *intellectual concerns* plays an important role in the validation and reuse of scientific workflows. Intellectual concerns consist of exploratory domain concepts, scientific models, representation of underlying theories and process specifications which form the basis for the scientific experiment and workflow. This information about the intellectual content of the initial formulation of a research problem will greatly assist *de novo* examination if the need arises for re-examination and reformulation of the problem. This examination can now be repeated by third party researchers by avoiding and excluding ineffectual workflow steps carried out by earlier researchers. This is also helpful in the investigation of alternative workflow scenarios.

In spite of the necessity of handling intellectual concerns in scientific workflow management, there is no indication of any ongoing efforts in this direction. Intellectual concerns are not handled well, if at all, in current scientific workflow management systems.

As observed in [Section 2.2.1](#), the focus on low-level engineering details in current practices of scientific workflows diminishes their ability to assist adequately in the *understanding* and *verification* of the rationale behind workflow orchestration and instrumentation. In addition, the lack of ability to directly map the underlying research principles to the workflow specification makes it difficult to validate the scientific soundness of the workflow.

2.3 IMPACT OF THE ABOVE ISSUES

The focus on low-level workflow engineering details also makes it difficult to rapidly adapt the workflow to changing problem conditions without the danger of inadvertent breakage. Consequently, debugging and repairing the workflow becomes error prone and time consuming. This is compounded by the open nature of science and scientific research, wherein newer streams of data are added from diverse sources faster than the ability of the scientist to analyze them comprehensively and effect changes and modification in the workflow specification to account for any anomalies, contradictions or unexpected new results [Steffen, 2008]. Additionally, the multidisciplinary nature of scientific research necessitates connecting together the knowledge of different disciplines. Approaches suggested for sharing concepts across different domains [Berkley et al., 2005] require defining significantly large sets of shared semantics. Thus, there is an absence of relatively localized ontologies, as a result of which, depending on the scale of the problem, the task of developing shared semantics can become intractable and unmanageable.

As disciplinary knowledge keeps on expanding, it becomes increasingly difficult to manage and sift through the huge amount of available information pertaining to a problem domain. This issue of scale and complexity of data management has been termed the *data 'deluge' problem* in literature [Deelman and Gil, 2006a; Deelman et al., 2009; Sahoo, Sheth, and Henson, 2008]. It becomes necessary and important to not only interconnect the vast 'silos' of information in each discipline, but also to adequately *preserve* and *exchange* the rationale of data and knowledge selection.

As a result of the deficiencies described above, scientific workflow modeling tends to become artificially deterministic and inflexible. Dynamic flexibility of data processing and analysis in scientific research is not available to the degree that would be desirable to ensure a highly robust and adaptive *in-silico* simulation of scientific processes. This is further compounded by data and process control issues which are routinely encountered in simulation science.

Feynman presents the following five criteria as essential for simulation of scientific processes [Feynman, 1982]:

1. Scientific and Physics computation and simulation data needs to be *locally inter-connected*.
2. In an *approximate simulation*, the algorithms used are approximations of physical systems to various degrees of semblance.

3. *Exact* simulations of physical scientific processes cannot be realized due to the unsatisfiable criterion to replicate a finite volume of space and time modeling the problem domain *exactly* within a finite number of logical operations.
4. Simulation of causality is at best a *partial imitation* of real systems.
5. As a consequence of the above facts, the probabilistic nature of physical experiments can only be simulated by repetition of a statistically large number of experiments.

Adopting the reasoning presented by Feynman, the modeling of scientific workflows and their results necessitates additionally stringent probabilistic requirements. Therefore, the scientific workflows need to be made adaptive enough to conveniently abstract and represent natural laws and processes which form the basis of research in a particular domain. This is necessary to understand and share the intellectual content of the scientific experiment.

Based on the above considerations, the following observations can be made:

2.3.1 Observation 1

Current scientific workflows are not abstract enough to capture the intent, rationale and intellectual concerns because they are heavily focused on low-level engineering details of execution and implementation.

2.3.2 Observation 2

It is not currently easy to record and encapsulate contextual information along with the scientific workflow specification. Underlying implicit assumptions are not known. This lack of information affects the ability to decide the appropriateness of the use of the workflow implementation or process for a specific purpose in a repeatable manner with proven validity.

2.3.3 Observation 3

Indeterminate provenance of the assumptions, data, context and theories underpinning the scientific workflow specification affects the confidence in the results and analysis provided by the scientific workflow management system.

2.4 CONJECTURE

In light of the above observations, it can be seen that in the current state of practice, even if a scientific workflow can be verified, i.e., checking that the workflow execution adheres to the workflow specification, there is little support for ensuring validity, i.e., ensuring that the proper workflow has been specified and implemented for the problem under study. So it is highly desirable to explicitly address these concerns in the management of scientific workflows. This can be achieved by:

1. Providing a layer of abstraction to lift focus from low-level implementation details.
2. Adding context support to scientific workflow components and participating processes.
3. Introducing localized ontologies applicable to a particular instance of a problem situation. This accommodates multidisciplinary interactions in intellectual concerns by significantly simplifying the process of arriving at a shared semantic understanding by reducing the scale and complexity of the ontological mapping. It is expected that localized ontologies will make it easier to effect changes owing to the resultant simplification in the management of the connections between the different disciplines involved in the workflow.
4. Providing an abstraction to capture and organize concepts, theories, ideas and rules (intellectual concerns) in the problem domain being explored and map them to the workflow specification and execution semantics.

This will ease the scientist's concern about data structures, formats and such low level-chores, enabling greater focus on the scientific issues of the research problem. As explained above, this will help clarify the rationale and intent behind the choices and decisions governing workflow specification, thus resulting in an improved understanding of the underlying concepts, models and theories. Also, reexamination and reformulation of the research problem becomes more tractable because of the enhanced information and support provided in the workflow.

The Omnispective Analysis and Reasoning (OAR) framework is proposed in order to address the above issues in scientific workflow management. OAR is conceptualized as an epistemic framework to better manage the intellectual concerns in scientific workflow management. The development of the OAR framework is intended to:

1. Raise the level of abstraction of scientific workflow management.
2. Enable support for enhanced context management.
3. Provide a mechanism to effectively capture and manage intellectual concerns.
4. Extend the scope and applicability of the scientific workflow process to other domains of exploration and research.

2.5 SUMMARY AND CONCLUSION

A historical overview of the evolution of scientific workflow management systems has been presented. Widely used systems like Kepler, Taverna and VisTrails have been briefly described. The main issues that are needed to be addressed in order to improve the management of intellectual concerns in scientific workflows have been identified. These have formed the basis for the observations and the conjecture as presented above, which has resulted in the research aim: *Formulate, develop and demonstrate an epistemic framework for managing intellectual concerns in scientific workflows.*

In [Chapter 3](#), the formulation and development of Omnispective Analysis and Reasoning is presented.

Part II

Omnispective Analysis and Reasoning

3

OMNISPECTIVE ANALYSIS AND REASONING

If science cannot provide an intellectual discipline for human activity, it is no better than mysticism.

Warfield [1994]

3.1	The Nature of Science	36
3.2	Fixation of Belief	38
3.3	Universal Priors to Science	39
3.4	Law of Triadic Compatibility	40
3.5	Hierarchy of Conceptualization	41
3.6	Domain of Science Model	42
3.7	Expanding Scale and Complexity of Scientific Work	44
3.8	Intellectual Concerns in Scientific Work	44
3.9	Defining Scientific Workflows	45
3.10	'Reforming' Scientific Workflow Management	48
3.10.1	Managing Intellectual Concerns	49
3.10.2	Dealing with Inadequate Context Support	51
3.10.3	Inadequate Support for Verification and Validation	51
3.11	Theoretical Foundations of OAR	52
3.11.1	Omnispective Analysis	53
3.11.2	Lifting Focus from Low-level Details	54
3.11.3	Defining Context and Adding Context Support	56
3.11.4	Localized Ontologies	58
3.11.5	Epistemological Basis	60
3.12	Overview of the OAR Framework	60
3.13	Prototypes, Archetypes and Constraints	62
3.14	Concern Refinement	63
3.15	'Bootstrapping' External Shelves	65
3.16	Context Refinement	66
3.17	Constructed and Organic Solution Specifications	70
3.18	Rationale for the Structure of OAR	71

3.19 Nature of the OAR Framework	73
3.20 Summary and Conclusion	74

The Omnispective Analysis and Reasoning (OAR) framework is presented in this chapter. This framework has been conceived as a means for providing effective management of all the concerns — particularly intellectual concerns — in scientific workflows. The structure of this chapter is as follows: As part of the key fundamentals in the development of the OAR framework, a brief exposition of the nature of science, the method of fixing belief, universal priors to science, the law of triadic compatibility and an outline of the Domain of Science Model (DoSM) are presented. A hierarchy of conceptualization is proposed in a form which is easy to comprehend and apply. The status of scientific workflows is revisited vis-a-vis the areas where there is need to focus attention and effect improvements as pointed out in [Chapter 2](#). An enhanced definition of scientific workflows is developed and presented in this discussion. The theory and foundations of the OAR framework and its structure and operation are presented. Within this chapter, all the terms and concepts used in the description of the framework are explained and the rationale for the particular choice of the framework structure is discussed.

3.1 THE NATURE OF SCIENCE

According to Chalmers [1999], science and scientific understanding encompass an epistemological discipline which consolidates the ability to effect interventions in problem scenarios of interest. Observations and understanding of physical phenomena lead to the design, development, construction and utility of practical applications and systems. For instance, analysis and understanding of electrical and magnetic forces has led to the construction of functional devices like generators and motors — devices that play an important role in daily life. Here it is interesting to note that the understanding of electrical and magnetic forces may not be final — it may evolve further as newer and newer experimental results are observed, resulting in improved understanding at a finer grain which may facilitate more and more esoteric applications (like lasers, which are based on the principles of electrical and magnetic interactions in atoms). However, refinement or reformulation of theories will not invalidate the applications that are based on earlier forms of the theory. Theories and hypotheses are simply constructs to analyze and organize the knowledge obtained through scientific studies in order to enhance comprehension and understanding. Thus, it is simply the basis of understanding that is altered and updated to be in tune with newly

acquired knowledge. This evolving nature is the essence of science and scientific understanding.

Sneider and Lerner [2009] observe that “science cannot provide unassailable proof that its understanding and interpretation of the natural world is correct.” For that matter, further reflection leads to the conclusion that unassailable proof of proper understanding is a utopia. The steps involved in the process of understanding can at best be demonstrated, the process itself being limited to the exercise of relating the phenomenon under study to more elementary *ab initio* concepts.

These steps in understanding may be considered to be a non-linear sequence of hypothesis–formulation, experimentation, observation and analysis. This is similar to the Observe–Orient–Decide–Act (OODA) loop given by Boyd [1986]. Though this concept has been originally formulated by Boyd in the context of devising and managing strategies in warfare, the loop consisting of Observation, Orientation, Decision and Action (OODA) models how a problem can be analyzed, organized and a response or solution designed and implemented.

Flint [2008] gives a customized version of the OODA loop in the domain of software and systems engineering [Figure 3.1](#).

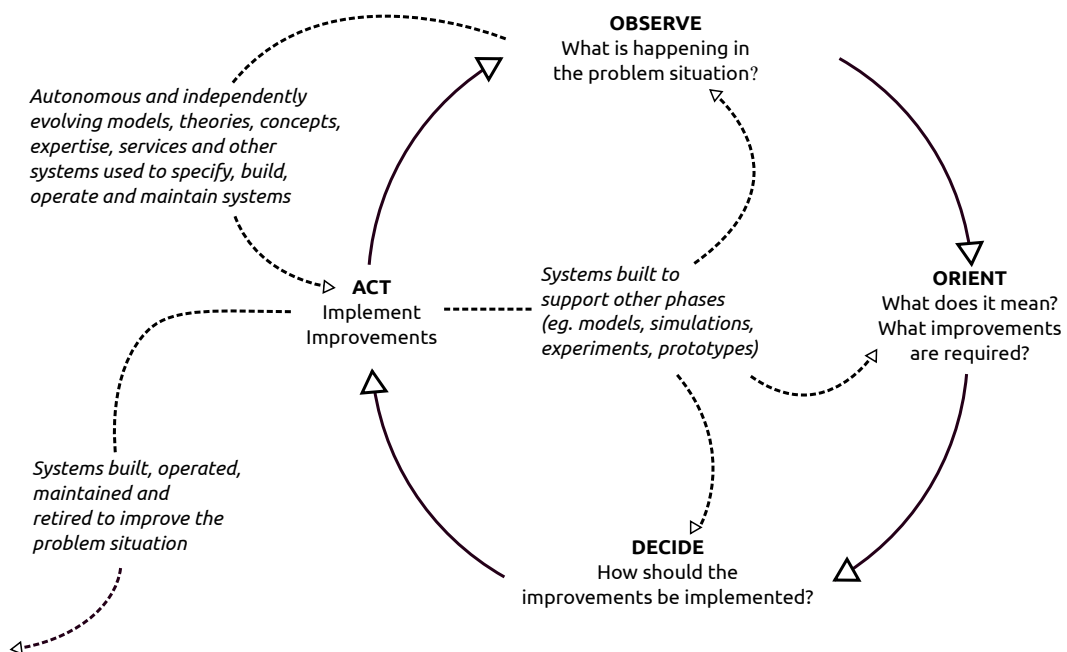


Figure 3.1: An adaptation of Boyd’s OODA loop for managing complex problem situations (after [Flint, 2008]). Copyright ©2008 Flint, S. Adapted with permission.

Here, the OODA loop can be considered as a model to represent the evolving nature of understanding a process by the method of science. The feature of

falsifiability of science leads to the process of modifying the basis of a science in light of new observations. This feature lies at the basis of the OODA loop.

As depicted in [Figure 3.1](#), the initial step of *Observation* involves examining the problem situation to identify its characteristic features that can be extracted and listed for further use. *Orientation* involves the analysis of the problem situation to look for possible areas of improvement. Interventions in these areas are planned during the *Decision* step — the nature, mechanism and execution details of the improvements are considered. Finally, the recommended improvements are implemented during the *Act* step. If at any stage, any of the steps are found lacking in some details, or newer data or changes in situation are encountered, the looping is carried out at each of the individual steps. Consequently, there can be multiple *micro/embedded* OODA loops at each step depending upon the granularity at which the problem is being analyzed. It may thus be noted that this has a parallel with the lifecycle of a scientific activity. Whereas a scientific activity mainly operates in the physical world with artifacts of instruments, observations and calculations, the adapted OODA loop operates with data, concepts, formulas, processes, computational procedures, etc. which may be either predefined or dynamically accessed.

3.2 FIXATION OF BELIEF

Before organizing and formulating the science pertaining to a chosen activity, the method of fixing belief and the priors of the science have to be agreed upon.

Fixing of belief means, starting from a state of *doubt* one would engage in a thought process to come to a state of *belief* — i.e., perceiving a feeling of certainty — on any given subject. According to Peirce [1877], four methods are employed to analyze, formulate and fix belief: the Method of Tenacity, the Method of Authority, the *a priori* Method, and the Method of science.

The Method of Tenacity consists of holding steadfastly to a belief as it used to exist and rejecting any and all beliefs that deviate from the one held previously. The process of forming a belief and sticking to it at all costs is sometimes mistakenly considered as a virtue of resoluteness. However, reflection shows that this corresponds to having a closed mind on a given subject. Adherence to this method results in dogmatism and retards the progress of knowledge and understanding. Also, in the long run, the dogmas may result in conflicts and confrontations with the beliefs of other people.

In the Method of Authority, the function of fixing belief is vested in an agent of authority who would decide which beliefs should be accepted and which

should be discarded. Others are expected to abide by the prescription and obey the authority. The process of arriving at the belief is not open for scrutiny or change. Amongst several drawbacks of this method, as pointed out by Peirce [1877], the possibility of institutionalizing and glorifying intellectual slavery is the most severe one and is detrimental to the progress of knowledge.

The *a priori* Method (also referred to as the Method of Metaphysics [Warfield, 1994]) requires that the fixation of belief be with reference to a system of thought or metaphysical philosophy propounded by one or more of thinkers and should be “agreeable to reason or harmonious and artistically pleasing.” The method does not insist that fixing belief should have basis in observed facts. It is like *ivory-tower thinking* and the beliefs arrived at may not relate to real-world situations.

The Method of Science requires that fixation of belief be carried out with reference to observed data and the verified and established laws of the science using repeatable and logical procedures. According to Peirce [1877], the fundamental hypothesis of the Method of Science is that the characteristics of an entity do not depend on any one’s opinions about the entity, and with sufficient observation and reasoning, these characteristics will be evidenced to be the same by different observers. The data as well as the process of arriving at the belief is open to scrutiny or modification depending on the developments in the state of knowledge about the given problem.

Out of these four methods, the Method of Science can be considered as fundamentally superior because it insists on validation/modification in terms of future experiences (as has been outlined in the operation of the OODA loop).

3.3 UNIVERSAL PRIORS TO SCIENCE

The *Law of Universal Priors* which has been established by the *Doctrine of Necessity* states [Warfield, 1994]:

The human being, language, reasoning through relationship and archival representation are universal priors to science.

No science can be formulated in any useful form unless these four prerequisites are ensured. In other words, a science is the result of human endeavor, expressed and communicated through a language, organized in terms of the relationships arrived at by reasoning (as provided by the procedures of logic) and archived for scrutiny and acceptance.

Knowledge of the characteristics of these priors gained from the studies in the relevant subject fields needs to be suitably incorporated in the design of a science to enhance the utility of the science.

Characteristics of the human being are taken largely from Social and Behavioral science. Language appropriate to the science may be adapted from mathematical and philosophical bases with elaboration of relevant meta-language (terminology) adequately in terms of natural language. Reasoning through relationship follows the processes of Analytical Philosophy. Archival representation uses terminology appropriate to the particular domain of study; it is necessary to avoid any miscommunication and terminology war by exercising restraint in using technical terms and taking the help of natural language to clarify whenever there is possibility of overloaded terminology across related branches of study. Being less elegant is better than being misunderstood.

3.4 LAW OF TRIADIC COMPATIBILITY

One important factor that needs to be given due consideration in the concept of any design is related to the limitation of the cognitive power of the human mind. Expressions like *foolproof* and *tamper-proof* design of even commonly used gadgets would suggest allowance for human limitations though this is not explicitly spelt out.

According to Warfield [1994], the capacity of human mind is limited; it cannot work with more than a limited number of concepts at a time. In this consideration, extremes of mental capacity are to be excluded. Warfield [1994] names this as a part of “bounded rationality” and emphasizes the importance of “finding ways to rationalize human problem-solving processes and thought processes within this limitation.” Based on several studies and rationalizations, Warfield [1994] proposes the Law of Triadic Compatibility as follows:

The human mind is compatible with the demand to explore interactions among a set of three elements, because it can recall and operate with seven concepts, these being the three elements and their four combinations; but capacity cannot be presumed for a set that both has four members and for which the members interact.

Thus, it is a highly recommended practice to structure a design such that at any instant at most three entities are presented for the mind to consider.

3.5 HIERARCHY OF CONCEPTUALIZATION

In this section, a hierarchy for the conceptualization of an entity is postulated in terms of three levels of conceptualization — Concept, Model and Implementation levels, with Concept placed at the highest level of abstraction.

The Data–Information–Knowledge–Wisdom (DIKW) hierarchy is widely mentioned in the literature of Information Science and Information Technology. It is generally attributed to R.L.Ackoff [Warfield, 1994], though the antiquity of such a concept has been pointed out in the literature (for e.g. see [Rowley, 2007]).

Ackoff suggests a hierarchy with five levels — Data, Information, Knowledge, Understanding and Wisdom and that the abundance of the entity decreases as one ascends the hierarchy. He contends further that these five entities are contents of the human mind and the entity at a lower level can be transformed to one at the next higher level by some kind of transformation. Warfield [1994] considers this hierarchy as a tautology. There has been considerable discussion and criticism of the hierarchy [Frické, 2009; Rowley, 2007], but there appears to be no clarification regarding the entities in the hierarchy or the transformations that change one into another. It has not been rendered into clear, simple and readily usable form. The scarcity of applications of the DIKW hierarchy indicates the possibility of the burden on comprehension due to including too many levels. Taking a fresh look from a different angle is needed for providing clarity.

The DIKW hierarchy hides more than it reveals. The terminology has not been explained and is subject to interpretation. When the entities are residing in the mind, it is not clear what is the meaning in saying that the abundance of the entity at one level is much less than the one lower. Data and information may have quantifiable features, but it is not so with understanding and wisdom. Sweeping statements like “more data than information” and “more information than knowledge” and pictorially representing with a pyramid may all be metaphysically appealing but do not help in either clarifying the idea or in enhancing the practical use of the hierarchy.

If a transformation renders the entity at one level into the entity at the next higher level, it is reasonable and logical to think that the various entities all represent the same *substance* at different levels of conceptualization or abstraction.

From the above observations, a hierarchy of conceptualization is proposed here based on the following premises:

1. It is the same entity or concern that is represented at different levels of the hierarchy. The different levels refer to different levels of abstraction.

2. Limiting the levels of the hierarchy to three will make the representation easy to comprehend and use.

A concern or entity (present in the study of a problem domain) can be abstracted at three levels — Concept Level, Model Level and Implementation Level (C–M–I).

The Concept Level represents the highest level of abstraction. Using analysis and reasoning processes to clarify the concern, it is abstracted at any of the three levels. Here the idea is not of transforming one level into another, but of conceptualizing the same concern at different levels of abstraction. The Concept Level abstraction represents the meaning and understanding of the entity as it exists in relation to the domain of study. The Model Level abstraction stands for the way the entity is represented in terms of simpler and related concepts and constructs of the domain. The Implementation Level considers the concern in terms of practical details like measurements, data collection, data processing etc.

3.6 DOMAIN OF SCIENCE MODEL

To bring out explicitly the mapping between the Content of a Science and the applications that are based on the Science, Warfield [1994] proposed the Domain of Science Model (DoSM). Its main features are:

1. Presents the knowledge embodied in a science and the methodology for representing the science.
2. Highlights the concepts and fundamentals that go into an application of the science.
3. Provides standards to ascertain the scientific nature of the contents and claims of the science.

The DoSM is constructed in such a way as “to describe what a science consists of, how its information must be organized in order to fulfill reasonable standards, and in what way a science must relate to its applications.” In this model, the domain of a science is divided into four functional blocks — Foundations, Theory, Methodology and Applications (Figure 3.2) where the blocks are connected to one another by a *steering relationship*.

The Foundations of the science form the basis for decision making and therefore should incorporate the universal priors as applicable for the particular science. The foundation should also contain the characteristic concepts of the science and the postulates that are essential for resolving issues in the theory and applications. A

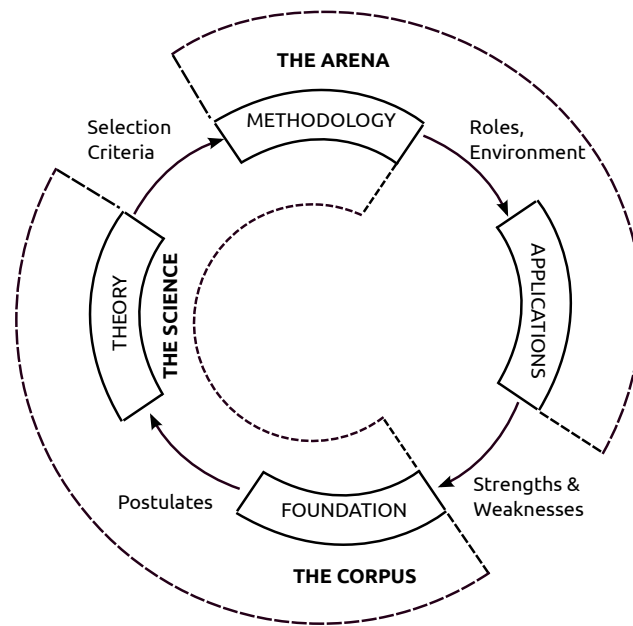


Figure 3.2: The Domain of Science Model (after [Warfield, 1994]). Copyright ©1986 International Society for the Systems Sciences. Adapted with permission.

well-formed Foundation is lean — it contains only those entities that can stand on their own and are established as valid and essential for providing referential transparency for their influence on the other components of the model.

The Theory block is meant to explain and establish the concepts and relationships in such a way as to demonstrate the integrity of the science as a body of coherent knowledge. It is very important to ensure that the theory provides “criteria for the acceptability of Methodology” and also should clearly define the conditions under which the Laws and Principles are applicable. This is necessary to guard against spurious methodologies and provide an “intellectual discipline” particularly in planning and designing large-scale systems where spurious methodologies may produce unexpected failures.

Methodology is mostly “prescriptive” and is intended to steer the Applications and may contain ideas, concepts, experimental procedures, process protocols etc. which are appropriate to the particular science.

All the applications supported by the content of the science form the Applications block. In the case of a complex application, a single science may not support it fully, and it may be necessary to combine a number of sciences in some fashion to do the job.

By organizing a science clearly in terms of blocks, the DoSM facilitates creation of a new integrative science by putting together in a compatible manner parts of a number of recognized scientific disciplines. The modular organization of DoSM, therefore, enables adequate management of complex problems.

3.7 EXPANDING SCALE AND COMPLEXITY OF SCIENTIFIC WORK

All except the simplest of scientific study involve a number of interacting disciplines and a variety of data logging, processing and management techniques. Increasingly ambitious planning in terms of scope and magnitude of projects often results in an explosive increase in the number of interacting concerns.

Nowadays, conceiving and planning a new scientific project is, in general, a complex exercise involving the interaction and ingenuity of a number of collaborating researchers, each having different areas of specialization, but all (presumably) geared to common interest in the progress of the research project. As projects are getting to be large and complex, lone researchers are becoming scarcer. A single person 'sitting under an apple tree' is no longer in a position to carry out experiments like the Large Hadron Collider (LHC) project. Research is becoming more and more collaborative and multidisciplinary, vast and distributed, requiring huge investments not only in terms of capital, manpower and equipment, but also in terms of ideas, theories, concepts, frameworks, methodologies, computational techniques etc. These resources may often be distributed geographically and additional requirements of communication, data transfer and synchronization need to be addressed.

Because of the huge 'costs' involved, every part of the project needs to be planned and executed with due care. All the data generated requires careful recording and evaluation for provenance, precision and relevance to the goals and objectives of the research program. This adds huge overheads of data book-keeping.

Scientific workflows come to the rescue of this kind of activity. They have been developed for the purpose of managing experimental and computational research, particularly for supporting large-scale data management, control and coordination of computationally intensive processes and the allocation, sharing and organization of resources in complex environments.

3.8 INTELLECTUAL CONCERNS IN SCIENTIFIC WORK

A variety of *intellectual concerns* are involved in scientific work. These include exploratory domain concepts, scientific models, representation of underlying theories and process specifications which form the basis for the scientific experiment and workflow. In this context, it may be mentioned that the term *intellectual*

concerns is not related in any manner to the term “intellectual property” often mentioned in relation to proprietary software systems and platforms.

In managing large and complex projects, considerable overheads of database management are added to the already present complex concerns of the research task. These additional overheads may not be intellectual concerns of the project; yet these require efficient management.

The theories and models that are used in a scientific investigation usually evolve from the past experience of science, settled principles and laws in the particular branch of investigation and the intuitive and analytical efforts of the practitioners planning the scientific investigation. For example, in natural sciences, entities like the laws of thermodynamics, Maxwell’s equations of electromagnetic waves, the electronic structure of atoms resulting in their chemical behavior etc. are considered as settled principles embodied in various laws. Models are built specifically to arrange and account for the results obtained by experimentation and observations. The models are based partly on settled principles and partly on hypotheses. Because of their very nature, the concepts and models keep on evolving as more and more data is examined or new situations are encountered in the experimentation. This requires that the data generated be properly identified, sourced and cataloged for use in data processing and computational activities of the research project.

3.9 DEFINING SCIENTIFIC WORKFLOWS

Scientific workflows as they are practiced today are software artifacts produced using the tools and techniques of computer programming primarily meant for aiding scientific work in the design, experimentation, data collection, data management and computation with facilities for presenting the results in a suitable format. Individual workflows may differ very much in scope, scale and complexity, but their main structure and ethos are essentially the same.

The following are three typical definitions offered in literature for scientific workflows as they are currently being used:

A scientific workflow is the description of a process for accomplishing a scientific objective, usually expressed in terms of *tasks* and their *dependencies* [Ludäscher et al., 2009].

We use the term *scientific workflow* as a blanket term to describe a series of structured activities and computations that arise in scientific problem solving [Singh and Vouk, 1996].

Scientific workflows are the formalization of the ad hoc processes that a scientist may go through to get from raw data to publishable results [Altintas et al., 2004].

While each of these definitions could be appropriate within the context in which they are presented, they are not of strong form and do not provide much information about the concept defined, and lack generality.

The term ‘scientific workflow’, for that matter, betrays a lack of precision in naming — it is primarily a workflow used for calculations and carrying out a sequence of steps during a scientific research project. However, such misnomers often get established in scientific literature from time to time — classic examples being: no reduction of efficiency in “reduced efficiency curves” of hydrocyclones, “fast breeder reactors” which actually do not breed fast, and many more such terms.

Essentially a workflow is intended to take care of the static and dynamic needs of a computational/procedural orchestration — either automated or manual — a set of procedures carried out in a logical sequence starting with a group of data as input and resulting in useful groups of derived data. In this light, since the term scientific workflow has been established by ad hoc usage, with researchers often talking about “typical scientific workflows” without providing any formal definition, or giving circular definitions like “a scientific workflow is a workflow used by scientists in their work,” it is necessary to formalize and expand the definition of a scientific workflow.

Since the main purpose of scientific workflows is to capture, model, implement and publish a particular scientific project, it is reasonable to suppose that a structure similar to the DoSM will be helpful in defining a scientific workflow.

According to Warfield [1994], “a Restriction that currently enjoys unwarranted favor is one that omits from the Domain of Science Model the Foundations and Theory, and includes only the Methodology and Applications. In this restriction, Science is cavalierly taken as synonymous with Methodology; or, alternately, one may say that the Arena is floating free of any scientific corpus.” Methodology is often dubbed as science and current workflow management systems do not appear to be any different in this respect. The emphasis and focus is more on the logistics of the workflow process than ensuring the appropriate processes are at play. This restricted (blinker) vision of workflow specification, orchestration and execution often obscures significant cues and feedback which can greatly aid and facilitate in better realizing the primary goals and objectives of the workflow exercise.

Hence it can be concluded that a ‘scientific workflow’ as currently understood and practiced does not qualify as science in view of the above considerations about the nature of science, nor does it correspond to the tenets of scientific investigation.

So, it can be reiterated that there is a need to broaden the definition of scientific workflow. It is desirable to base this new definition on the idea of the science of design of large-scale systems and the method of science for fixing belief to make it more inclusive of the four universal priors of science.

The definition of a scientific workflow should encompass any process from the simplest to the most involved. Examples of scientific workflows can range from the most trivial to the most complex — such as opening a bottle of jam, investigating the interactions in a chemical reaction and building and operating a nuclear power plant. A broadened definition of a scientific workflow should be able to encompass all these cases. So, circular definitions are not the answer. The definition should reflect the basic nature of science.

It is instructive to inquire whether a certain activity which is not commonly described as a scientific activity can yet be analyzed, modeled and orchestrated by a scientific method. Reflection tells that the answer is yes. Once a basis is assumed and data, parameters, processes and interactions are specified, application of the scientific method will enable intervention in the problem situation irrespective of whether the problem is initially considered a ‘scientific problem’ or not. Therefore, the definition or concept of a scientific workflow need not be tied down to structured steps of scientific objectives.

The application of the scientific method can be suitably adapted to give a well-rounded definition of the term scientific workflow. The nature of scientific method incorporates logical, systematic and repeatable inquiry. Hence the following definition of a scientific workflow can be arrived at:

DEFINITION 1 *A scientific workflow is a representation of any logical, systematic and repeatable inquiry, investigation and corresponding set of actions.*

The inquiry and investigation represented by a scientific workflow is logical, i.e., derivable from a set of consistent and valid premises. In other words, it has basis in the Foundations of a Science. Inquiry and investigation cover exploratory research as well as detailed in-depth studies.

The set of actions are the various steps involved in the formulation and implementation of the workflow and are required to be logical, systematic and repeatable (and may be ad hoc or structured).

The results obtained from the execution of the workflow need to be of repeatable nature. If variability in the outcomes is inherent in the nature of the investigation, repeatability has to be in the sense of such a behavior.

Based on the criteria outlined by Warfield, it can be seen that Definition 1 is a Definition by Relationship [Warfield, 1994] and in addition to incorporating the traditional view of scientific workflows, it also extends the scope of ordered

analysis and investigation to any generic problem scenario. Any workflow that has been specified in terms of a well–formed structure consisting of Foundation, Theory and Methodology comes under this definition.

3.10 ‘REFORMING’ SCIENTIFIC WORKFLOW MANAGEMENT

Issues that require to be addressed to improve the management of intellectual concerns in scientific workflows have been identified in [Chapter 2](#). However, definite proof of the existence of these needs cannot be adduced owing to the peculiar nature of the practices of scientific workflows. Extensive literature search does not reveal much progress in these directions — even when mentioned, these ideas have been considered in a cursory manner only.

In order to establish the case for the need of effective management of intellectual concerns, the line of reasoning presented by Warfield [1994] in the context of developing new approaches is employed.

1. “Most–usedness” is not necessarily completeness of a design. A few areas of excellent application and useful results do not obviate the need to make the system applicable to generic cases.

In dealing with scientific activities, it may so happen that intuitively felt procedures are tried out and often with useful outcomes. This idea is supported by the contentions of Sneider and Lerner [2009]. The merit of such ad hoc procedures consists of expediency in obtaining useful results. However, there is no forthright acknowledgement of such expediency in the documentation of the activities. Therefore, these should not be construed as evidence of systematic, well–designed and well–understood practices for managing intellectual concerns.

2. Big projects and great names should not deter from attempting fresh thinking and making structural changes in an attempt to better an existing practice.

Occasionally, large and complex socio–technical problems surface, and these are managed by experts using ad hoc ingenious methods. However, these are not intended to provide a systematic approach to problems of that type, since they are mainly focused on mitigation of a particular instance of a problem situation. Therefore the original issue of providing a systematic formulation of solution methodology is not touched upon, and it still remains an open research problem. Various large–scale projects have been undertaken with specific goals of promoting exchange and sharing of

scientific workflow specifications and artifacts [De Roure, Goble, and Stevens, 2009]. However, their focus and aim is not on enabling the identification, capture and management of intellectual concerns. Consequently, the need for addressing the issue of intellectual concern management still exists and requires a novel perspective and approach.

3. Once the need for improvement in current practices is felt, efforts should not be frittered away on simply providing “cosmetic treatment where a heart transplant is required.”

Approaches geared to ‘tinkering and tweaking’ existing practice often run in the groove and may not produce complete remedies. Attempting a few touches of refinement here and there in existing workflow management systems will not provide much success in addressing the issue of managing intellectual concerns since the requisite scaffolding to support the management of intellectual concerns is not in place. A new direction and approach is likely to produce more fruitful results.

Based on the line of reasoning presented above, the issues identified in [Chapter 2](#) are discussed further.

3.10.1 Managing Intellectual Concerns

It may be argued that there is no special need for making the effort to manage intellectual concerns, given that current scientific workflows with the existing emphasis on low-level details still manage to produce useful results. However, the fallacy of this argumentation becomes apparent in light of the visible increase in the complexity of problems encountered by scientific researchers. Left to its own means, the present approach of tackling problems exclusively at low levels of data and process management will add to the difficulty of managing the issues of scale and interaction.

Hence it is desirable to introduce some discipline in this seemingly intractable complexity by ‘stepping back’ in order to ensure that the intervention made and resultant workflows still remain relevant in the larger context of multiple interacting disciplines. In addition to having the ability to tackle low-level protocols of experimentation, research, engineering and organization efficiently, it is also necessary to develop the capability to interpret and evaluate individual workflow outcomes under the lens of a wider problem context. In order to realize these gains, providing a basis for understanding of intellectual concerns in the workflow has to be *deliberate* rather than *fortuitous happenstance*.

Thus, even though current workflow management systems may make it appear that managing low-level concerns will completely enable robust scientific workflows from conceptualization to implementation, it is not always so, because suitable abstraction for mapping the workflow implementation to the underlying theories, models and data is not present.

As seen in [Chapter 2](#), existing workflows are mainly utilized for flowcharting control of experimental data collection and processing data to produce computed results or making database comparisons. Applications like pattern matching — in genetic code identification — demonstrate the usefulness of existing scientific workflows in managing huge quantities of processed data. This in itself is sometimes considered as the end goal of scientific workflows. Although this is a commendable objective of scientific workflow management, explicit provisions for the analysis, verification and validation of theories, models, concepts, computational algorithms, approximations and assumptions are not adequately focused upon. These may often be lost sight of, creating the illusion that great breakthrough in scientific research has been achieved, indirectly resulting in cost overruns if it becomes necessary to modify or correct some of the above *intellectual concerns* at a later stage of the investigation. Even though mechanisms for addressing and managing intellectual concerns are not currently available, the usefulness and need of such mechanisms in order to realize the goals and objectives of scientific research cannot be overemphasized. Issues encountered in managing the data ‘deluge’ of scientific computations should not be allowed to drown the spirit of scientific investigation.

From the abacus to the supercomputer there has been a steady evolution in the computational capability of digital machines. Digital computers operate by manipulating numbers. Data processing and data management are an essential and inalienable part of any scientific workflow. However, it is the ingenuity and intellectual efforts of scientists that shapes the manipulation of numbers into useful and powerful programming concepts that form the basis of data processing techniques and tools. Thus, adequately characterizing all the relevant intellectual concerns will greatly enhance reliability and repeatability, in addition to infusing resilience in modifying and correcting underlying concepts, theories and models. Adding explicit features for managing intellectual concerns in a scientific workflow can thus be viewed as providing enhancements that are very much in tune with the basic tenets of computational science. This is now in the realm of practicability, thus strengthening the claim for giving prime consideration to intellectual concern and associated intellectual effort management in scientific workflows.

3.10.2 Dealing with Inadequate Context Support

In addition to the focus on low-level implementation details, it has also been pointed out in the previous chapter that no significant attempt seems to have been made to define context and use it as a specific feature of scientific workflow management.

The word 'context' literally means "the set of facts or circumstances that surround a situation or event." It may also have shades of other meanings depending on usage. However, though the terms 'context' and 'context-aware workflows' have been applied in various ways in relation to significant contributions to scientific workflows [Chin Jr. et al., 2011; Ngu et al., 2010, 2011], they have been used in the sense of a scope or boundary of execution of a process, personnel or resources involved in a process — their ranks and designations, data formats like whether something is integer, float or scientific notation and the when-where-what-who have been involved in a process. Thus, context is considered more like historical documentation. Such historical information has its own merit and usefulness and all data sets and databases should be labeled with such information to keep their usefulness and integrity intact. However, with reference to the OAR framework, context is defined as *the relationship of one particular entity and its influence on other entities occurring in the workflow (scientific) process*. An entity here is not the low level format of the data or a database record etc.

A formal definition and model of context for entities in a scientific workflow is provided in [Section 3.11.3](#).

3.10.3 Inadequate Support for Verification and Validation

As emphasized above, the focus on low-level implementation details and lack of proper support for context in the sense as elaborated, result in inadequate support for verification and validation of the underlying science of the workflow. Such verification and validation cannot occur unless suitable mechanisms based on analysis and reasoning are deliberately provided in the formulation of the workflow. In other words, there is inadequate support for the management of intellectual concerns in the existing practice of scientific workflows.

Reasoning in the above fashion, it can be concluded that developing a suitable framework for managing intellectual concerns in scientific workflows is highly desirable. Accordingly the idea of Omnispersive Analysis and Reasoning (OAR) has been developed as an epistemic framework for managing intellectual concerns in scientific workflows. The concept and scope of the OAR framework is defined

in Figure 3.3. The extended definition of a scientific workflow, developed earlier in this chapter, is used in this framework.

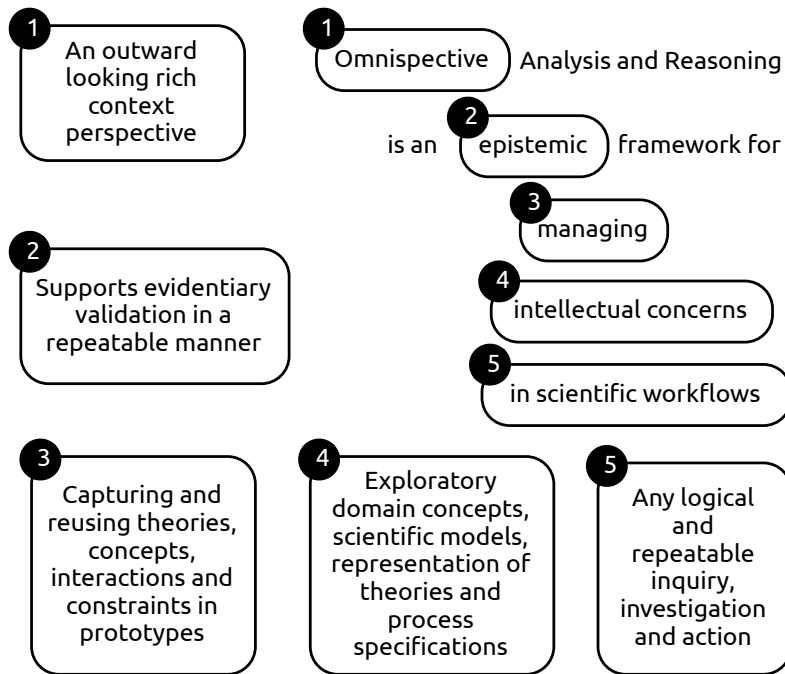


Figure 3.3: A definition of the OAR framework.

3.11 THEORETICAL FOUNDATIONS OF OAR

The theoretical foundations for the design of the OAR framework to manage the concerns — particularly the intellectual concerns — in scientific workflows, are presented in this section. The issues that have been identified earlier can be addressed by:

1. Introducing the concept of Omnispection coupled with the logical processes of Analysis and Reasoning. This ensures that all available knowledge (relevant to the given problem) is included in the workflow and areas relevant but lacking information are identified.
2. Adding a suitable layer of abstraction to lift focus from low level implementation details.
3. Adding context support to the workflow components and participating processes; an appropriate model of context is defined and used for this purpose.

4. Introducing the idea of localized ontologies applicable to a particular instance of a problem situation.
5. Providing an epistemological basis to consolidate the knowledge incorporated in the workflow in order to capture and organize the intellectual concepts, theories, ideas and rules in the problem domain being explored and map them to the workflow specification and execution.

While the processes of context support and localized ontologies are akin to “information operations” [Warfield, 1994], the framework provides additional support for the intellectual burden of dealing with data structures, formats and low-level chores, thus enabling greater focus on the scientific issues of the problem. This not only helps clarify the rationale and intent behind the choices and decisions governing workflow specification, but also leads to an improved understanding of the concepts, models and theories involved.

The foundations of the framework have been conceptualized in light of the basic considerations which have been presented earlier in this chapter. Almost all scientific workflow systems are large scale systems and therefore it is important to render the design easy to understand, comprehend and utilize in a given problem. For instance, the *Law of Triadic Compatibility* is used as a general guide (limiting hierarchical arrangement to triads wherever possible) to conceive and realize the foundation and framework, respectively.

3.11.1 Omniscpective Analysis

Omnispection means “looking at All of It” — examining all that can be discerned and gathered about the problem under study.

Analysis by division, which is attributed to Plato and Aristotle, is a powerful process to ascertain and bring out the characteristics of a complex concept and provides a means for defining the concept. Analysis consists of dividing a concept into a set of constituents, second level concepts, and then to proceed to divide each of these into smaller parts and continue with the process until no further clarification results by further division.

Reasoning is the process used for discovering the relationships between different entities and forms the basis for drawing inferences and making decisions. Reasoning, at a formal level, is considered as part of Analytical Philosophy.

Application of Omniscpection, Analysis and Reasoning, results in assembling all the relevant information of the inquiry domain abstracted into convenient units (concerns) encapsulating clearly all the relevant interrelationships forming the overall repertory from which one can define the scope of the problem and

formulate a solution. This is similar to the concept of universe or universal system in thermodynamics, where the universe pertains to the entirety of the problem under study, while extracting concerns is like defining unit processes in the study of complex chemical engineering processes.

Following the analogy of unit processes in chemical engineering, and to preclude any possible confusion with ‘concern-oriented’ software development, a concern is designated here as a Unit knowledge entity (uke). A uke is a certain amount of knowledge and information which has been abstracted into a convenient form to bring out its relationship to the problem under study and to other concerns in the problem.

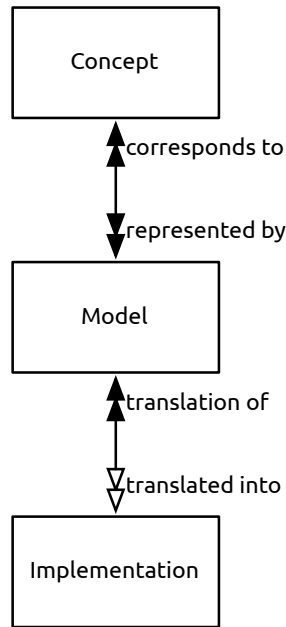
In carrying out the omnispersive analysis, the problem needs to be examined from all angles of all the interacting disciplines that are considered relevant. Iteration steps as explained in the OODA loop are employed during the process to achieve well-formed uke groups to represent the problem in its entirety. At this stage, it will be possible to identify areas considered relevant but not adequately enunciated or recorded — these can then be marked as areas needing further study and experimentation. Some more benefits of the analysis and abstraction processes are identifying ukes that are critical to the problem, revealing the precision of data and recognizing the possible interactions between different sub-domains to avoid unexpected results. This capability of omnispersive analysis would be significant in analyzing and understanding the emergent behavior of large and complex systems.

3.11.2 Lifting Focus from Low-level Details

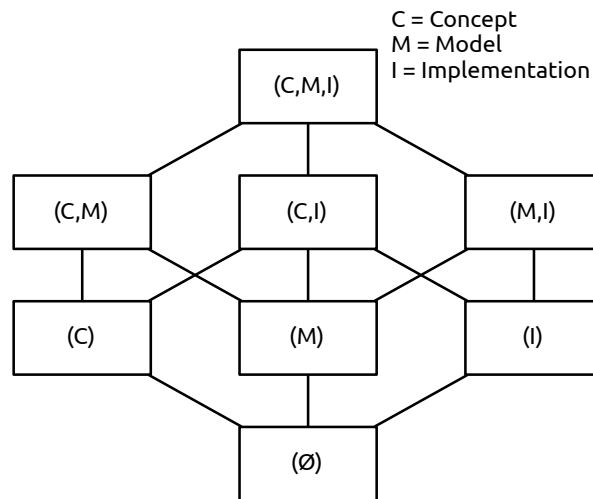
In current practices, scientific workflows are specified and constructed in terms of “processes, tasks and their dependencies” [Ludäscher et al., 2009] or as “structured activities and computations” [Singh and Vouk, 1996]. The focus is mainly on the implementation details and is divorced from the scientific context of the problem.

In the approach presented here, the lifting of focus from low-level details is achieved by adding a layer of abstraction to the process of analysis. For this purpose, a uke is considered at three levels of abstraction — concept, model and implementation levels, which have been discussed in [Section 3.5](#). A uke may encapsulate concept, model, implementation or a synthesis of any combination from the three as shown in [Figure 3.4](#). The abstraction process effectively ‘separates’ the implementation details from the entity and enables capturing and mapping of the underlying science to the workflow process.

The ukes are collected and managed at three levels in the hierarchy ([Figure 3.5](#)). At the concept level, the ukes include exploratory domain concerns and the



(a) Representation by XTUML (relations).



(b) Representation by lattice (states).

Figure 3.4: Composite nature of Unit knowledge entity (uke).

interactions and constraints between them. At the model level, they represent physical and logical systems in terms of formalisms incorporating theories and paradigms. These may be abstracted in mathematical and analytical models, vocabularies, data sets, natural language representations, ontologies and process guidelines. Implementation level entities capture system frameworks, mechanism and translation processes to satisfy specifications in terms of and conforming to the ukes at the concept and model levels.

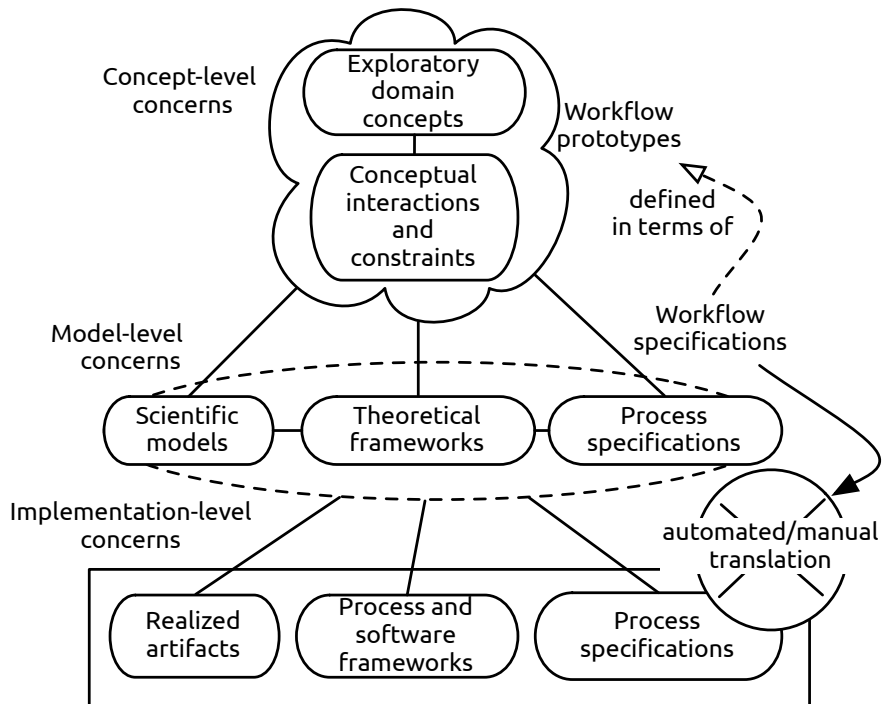


Figure 3.5: Hierarchy of concerns

Every uke within the repertory can be evaluated at any of the three levels depending upon the context of the problem analysis. Depending upon the outcomes of the analysis and the relevance of the uke to the problem under study, it is considered at an appropriate level of abstraction.

Thus operation of the layer of abstraction will effectively 'separate' the implementation details and enable focus on the underlying science.

3.11.3 Defining Context and Adding Context Support

In [Section 3.10.2](#) the context of an entity has been considered as the relationship of the entity and its influence on other entities in the scientific problem domain. Here, the well-formedness of a uke or group of ukes and its *influence* on another

is referred to as context. Additionally, the idea of describing entities by attributes characterized by profiles [Warfield, 1994] is interwoven into the definition; these attributes may be either quantifiable or qualitative parameters judged by gradation. In using attributes that have connotations of notional, perceptual or judgmental character, it is necessary to anchor the description and use of the attributes to referential transparency encapsulated in the method of science.

In the present study, the interest is to employ *context* which is clearly specified in terms of a set of attributes or dimensions in such a way as to bring out the soundness and relevance of the intellectual concerns embodied in the ukes as also to establish their role in the workflow formulation and solution.

Here, context is considered as a function of two dimensions (attributes) — Firmness and Influence. A graphical representation of the context of a uke is shown in Figure 3.6. In Figure 3.6a the two dimensions I and F characterize the context of the uke. The same is shown by the profiles of the two attributes in Figure 3.6b.

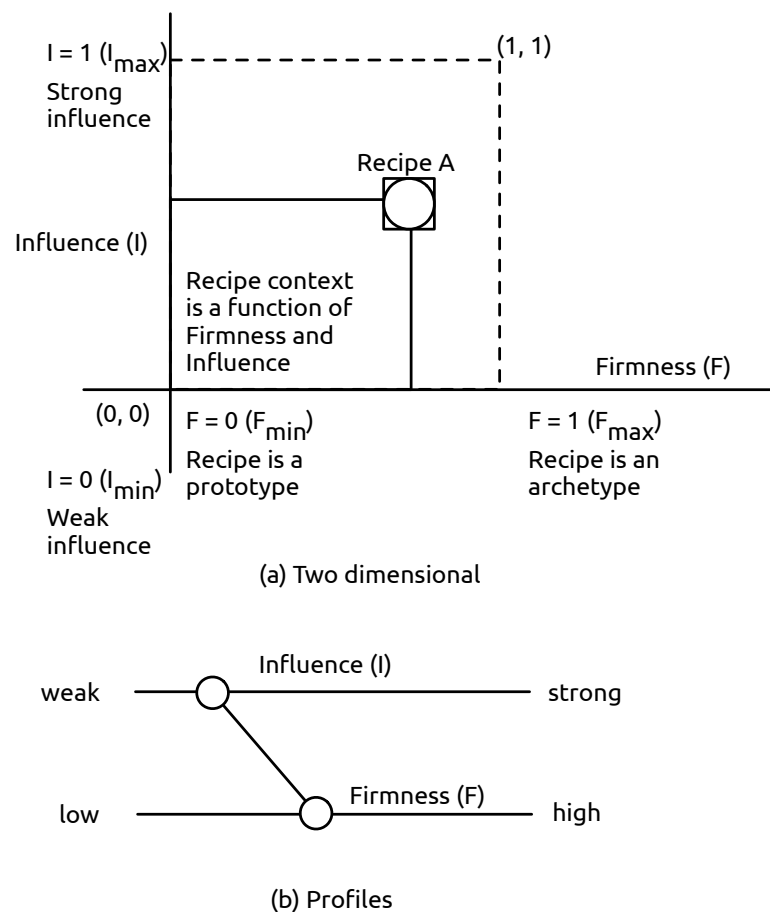


Figure 3.6: Uke context as a function of Firmness and Influence.

Firmness is a measure of the degree of well-formedness of a uke. If a uke is ambiguously defined, vague, or peppered with implicit characteristics, the knowledge encapsulated therein can be considered to be pliable and has *low firmness*. On the other hand, a well-formed and explicitly defined uke has *high firmness*.

Influence is a measure of the effect exerted by a uke in the analysis of the problem domain. If an entity is identified as relevant to the problem domain, then it exerts a non-trivial influence. If it exerts a strong influence, then it is identified to encapsulate *exemplar criteria* or best practice for the problem situation. If it does not affect the problem domain, but may still be considered relevant, it is identified as exerting weak influence.

To illustrate firmness, consider a uke T representing the temperature of a room. If T is characterized as *warm, a bit hot, or rather cold*, it is vague and hence the firmness of T is low ($F = 0$). Instead, if it is specified as $T = 301\text{K}$, the firmness of the uke is high ($F = 1$).

To illustrate influence, consider the following three ukes.

A: rise of temperature from 301K to 305K

P: volume of a room

Q: *state of aggregation* of a sample of gallium (Ga) kept in the room

It can be easily reasoned that the influence of A on P is negligible ($I = 0$). On the other hand, influence of A on Q is quite significant because gallium melts at 302.8K and changes from solid to liquid state ($I = 1$). The context $C(I, F)$ for these two cases can be denoted by the relations:

A $C(I = 0, F = 1)$ P

and

A $C(I = 1, F = 1)$ Q

3.11.4 Localized Ontologies

The concept of localized ontologies introduced here, is applicable to a particular instance of a problem situation. This will facilitate the process of arriving at a shared semantic understanding and reduce the scale and complexity of the ontological mapping. This will also make it easier to make changes owing to the simplified management of the connections between different disciplines involved in the workflow.

The increasing heterogeneous and multidisciplinary nature of scientific processes generates new challenges in interconnecting the knowledge of the domains involved. Each discipline has its own deep 'silos' of knowledge with its own terminology, notations, conventions and semantics. These need to be connected

together. This is not trivial and puts forth issues of common agreement of terms, data presentation and formats to various degrees of tolerances. To take a simple example, this scenario is routinely encountered when comparing a modern data set having standardized metrics with a historical data set where no strict adherence to one set of units or conventions is maintained.

Although various ontological approaches have been suggested to address this problem by forming and managing a shared set of concepts across disciplines [Berkley et al., 2005], in these approaches, the degree of ontological formalism necessitates *a priori* agreement on a significantly large set of shared semantics before the ontology can become useful. While it is necessary and important to interconnect vast ‘silos’ of information in each discipline, it is also important to adequately preserve and exchange the rationale of data and knowledge selection for the problem under consideration.

A localized ontology is realized by relaxing the requirement of *exhaustiveness* in the ontology. Only *satisficing knowledge* is encapsulated in the ontology which is applied to the problem workflow, even if it may lack universal applicability. For instance, planetary motion is analyzed in terms of classical mechanics, while electron motion in atoms is analyzed by quantum mechanics. Although not explicitly labeled as such, this illustrates the concept of localized ontologies. Thus, depending on the scale of the problem, localized ontologies are necessary to make the task of developing shared semantics manageable and tractable.

The concept of localized ontologies springs from the logical process of *analysis by division* and has been in use in some form or other in the documentation and discourse of scientific knowledge. The archival exposition of human anatomy [Gray, 1918] can be considered as an outstanding application of localized ontologies. The anatomy is considered as a collection of several systems; each system is described separately, keeping in view the interactions with other systems. This clearly illustrates the role and usefulness of localized ontologies in analyzing, understanding and managing a large and complex system.

In the domain of software engineering, enhancing the flexibility and comprehensibility of software systems by means of modularization [Parnas, 1972], separation of concerns in software development [Dijkstra, 1982], partitioned iterative approach for speeding up development of “Large Complex Business Management Systems” [Sessions, 2006] and Aspect-Oriented Thinking process of dividing a problem situation into a number of “autonomous domains” and describing them by “Domain Models” [Flint, 2006] represent ideas similar to the concept of localized ontologies.

3.11.5 Epistemological Basis

To ensure the evidential validation of the ukes and the workflow intent and rationale, there is a need to provide explicit epistemic hooks in the workflow management. An epistemic hook is a deliberate provision for encouraging the practitioner to ensure that a given concern encapsulates the necessary information regarding its applicability, correctness and completeness for a given problem scenario.

Incorporating hierarchical management of ukes enables the capture and cataloging of workflow concerns in a repeatable manner. This also facilitates end-to-end mapping of the underlying theory and data in the scientific analysis to the workflow execution.

The processes of concern refinement and context refinement are incorporated into the operation of the framework to ensure verification, validation and mapping of the relevant ukes to the workflow and solution specification. These are elaborated further in sections 3.14 and 3.16.

3.12 OVERVIEW OF THE OAR FRAMEWORK

Designing a framework involves considering the options possible and organizing them to realize the purpose of the design. The OAR framework is designed based on the theoretical foundations presented earlier in this chapter.

The basic concepts of the DoSM and the Science of Generic Design are kept in view in constructing the OAR framework. Since the scope of scientific workflows may range from 'fairly simple' to 'highly complex' scenarios, the design has to be such that, even in the case of a highly complex scenario, it should be easily understood, comprehended and utilized. For this purpose the Law of Triadic Compatibility (Section 3.4) is used as a guide. This law has been demonstrated to be effective in producing design of complex systems in such a way as to avoid unexpected difficulties and failures — by bringing all identified concerns in the problem clearly into the compass of comprehension of practitioners designing and using the systems.

As seen in Chapter 2, scientific workflows can be categorized as exploratory research workflows or service-oriented workflows.

Workflows used for furthering understanding and the search for scientific knowledge by utilizing experimentation, data collection and processing are characterized as exploratory research workflows. These comprise of instances of workflows ranging from concept-level to implementation-level and involve

concerns which may be exploratory concepts, assumptions or hypotheses; these may be modified depending on the outcomes of the workflows [Baker, McClatchey, and Le Goff, 1997].

In service-oriented scientific workflows, a settled set of formulas, principles and processes are utilized to produce data, computed results and inferences which are put to practical use. These typically correspond to workflows at the implementation level. For instance, scientific workflows which are designed for gene pattern matching in bioinformatics that involve processing huge amounts of data using a selected set of computations; medical investigation by analyzing blood samples, CAT scans etc.; financial instrument processing by banking institutions; and grading student performance in standardized tests.

In addition to the above two categories, there may also be instances where an investigation has features corresponding to a scientific inquiry, but the adjective 'scientific' is not explicitly applied. One can find a number of examples for such scientific activities from various domains like mathematics, humanities, economics, applied arts etc. On the other hand, there may be instances where certain domains are referred to as 'sciences' even though they may not stand the test of referential transparency. These fall outside the scope of systematic and logical investigation and are not considered in the present study.

Based on these considerations, OAR is designed as an epistemic framework for managing intellectual concerns in scientific workflows. The definition illustrated in [Figure 3.3](#) states the scope and approach of the framework. The scope covers the study and management of any generic problem situation that can be represented and analyzed in terms of logical and repeatable steps and procedures. This follows from the enhanced definition of scientific workflow given in [Section 3.9](#), and also from the epistemic nature of analysis and reasoning combined with the process of abstracting concerns as ukes at concept, model and implementation levels. This in turn, results in revealing the intellectual concerns and contextual relationships between various entities. Additionally, this encourages the researcher to verify and validate the problem and solution specification with reference to the scientific basis of the research domain containing the problem scenario.

A problem scenario in a scientific investigation generally consists of numerous interacting concerns. All concerns that are identified as relevant and likely to influence the outcomes of the scientific investigation are considered for formulating and implementing the workflow. Hence the framework is intended to enable an *omniscient* appraisal of the scientific investigation, providing an outward looking rich context perspective for analyzing and reasoning about the underlying concerns in the problem situation. These concerns are abstracted as ukes and formed into convenient groups called recipes. Based on consideration of their

relevance and well-formedness, recipes are categorized as *prototypes*, *archetypes* and *constraints*. These, collectively, represent all the information available about the problem scenario. These are managed in *shelves* which are unordered collections of recipes. Three categories of shelves are used — *external*, *problem-domain* and *solution* shelves. All known recipes from different domains are held in external shelves. Recipes that satisfy problem criteria are selected by the process of *concern refinement*; these are held in the problem-domain shelf. In a solution shelf, a specification of the solution is obtained by analyzing the prototypes, archetypes and constraints in the problem-domain shelf using *context refinement*. If the solution specification requires further translation, the OAR processes are carried out again to formulate an implementation in terms of available implementation-level recipes.

3.13 PROTOTYPES, ARCHETYPES AND CONSTRAINTS

In an earlier section, the term uke was defined as a unit knowledge entity abstracted during the process of omniscient analysis. Following the general usage of language, the terms ‘concern’ and ‘uke’ were used as interchangeable. However, for the purposes of clarity, the terms ‘concern’, ‘recipe’ and ‘uke’ will be used with their meanings related as shown in [Figure 3.7](#).

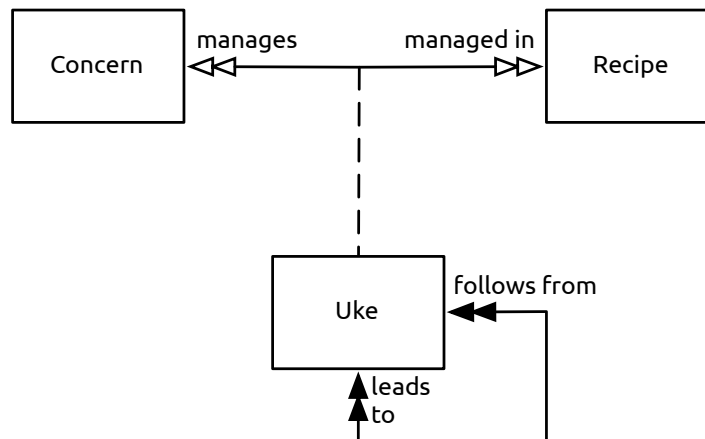


Figure 3.7: Relation between concern, recipe and uke.

A recipe consists of one or more ukes (that are related and interacting) which will be directly useful in the formulation of the workflow specifications. Recipes are formed to facilitate a clear and repeatable representation of the problem

scenario. Concerns are related to the problem being investigated. Recipes are useful in managing the concerns.

Depending on the degree of Firmness and Influence, OAR recipes are categorized in three types — Prototypes, Archetypes and Constraints.

A Prototype is any OAR recipe that is available in a given domain without particular consideration of its applicability, degree of formalism or robustness for any fitness or purpose. Thus, a prototype can encapsulate either nascent or well-formed domain concerns that may be available to support the analysis of any problem situation using the OAR framework. Depending on the area of study, the prototypes can range from rudimentary outlines and sketch-ups to formal blueprints.

An Archetype is a prototype that may be considered an *exemplar* or *best practice* recipe for a concern in a particular domain. The choice of an archetype is dependent upon the analysis of the problem under study and influences the net understanding of the problem domain.

A *constraint* is a special instance of an archetype which is identified as imposing strict criteria on an OAR specification. A *valid solution* of the problem is required to *sufficiently satisfy* all the requirements of the constraint without exception, and is often subject to strict conformance.

3.14 CONCERN REFINEMENT

Individual concerns (ukes and recipes) in OAR are managed in collections known as a *shelf*. A shelf is simply an unordered collection of recipes categorized by individual subject domains and their relevance to the problem under consideration within the OAR framework.

Each OAR shelf can accommodate zero or more prototypes. A prototype need not necessarily be an atomic uke and overlapping prototype groups can exist depending on the granularity of the domains under study. The distinction between individual prototypes is ultimately dependent on the context of the problem situation. Shelves can be further categorized into *external*, *problem-domain* and *solution* shelves (Figure 3.8).

Various *External Shelves* hold all the known recipes (prototypes) from different domains of interest in the analysis of the problem situation. Each external shelf is populated with concepts, data, constraints, models, details of data collection procedures, and experimental processes — in a reasonably usable form. Any number of external shelves can be populated to accommodate all the recipes.

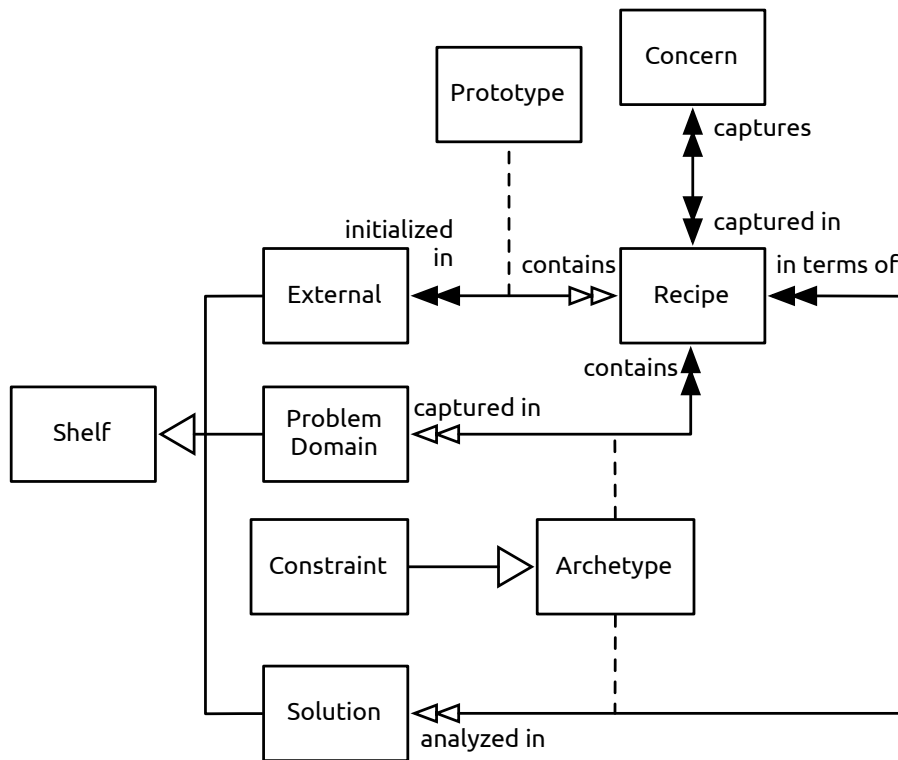


Figure 3.8: Shelves and recipes.

The *Problem domain shelf* holds recipes selected from various external shelves that satisfy given criteria in the problem under consideration. These exemplars or *best practice* recipes are effectively *Archetypes* in the problem domain.

The *Solution Shelf* consists of recipes which are specifications of the *archetypes* in the problem domain subject to *Constraints* that are identified and imposed on the particular instance of a *Solution*.

Depending on the context, a solution shelf may either be an executable domain or may require further translation.

Concern refinement is carried out in the following way, as illustrated in [Figure 3.9](#):

CNR-1 Initialize and populate external domain shelves with prototypes. This bootstrap step is not required if domain knowledge is available in identified preexisting external shelves.

CNR-2 Identify prototypes and possible archetypes and constraints from various external shelves which have a bearing upon the problem under consideration. Collect these recipes in the problem domain shelf. All recipes identified in this step need not be utilized in the solution shelf.

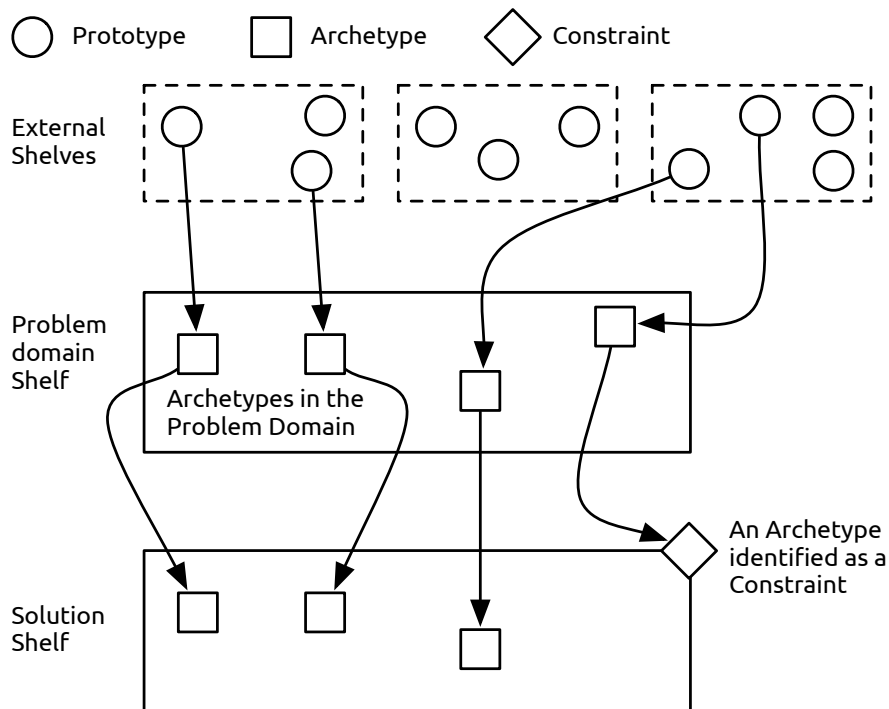


Figure 3.9: Managing concerns in the OAR framework.

CNR-3 Analyze the archetypes in the problem domain using *context refinement* to specify a *solution* specification in the solution shelf. This step also identifies the problem domain archetypes which impose limiting criteria on the specification and are termed as *constraints*.

The OAR framework facilitates localized ontologies via shelf management. The external shelf need not represent a formally complete understanding and representation of all domains that are cataloged in the framework. The problem domain shelf helps to build a localized ontology which is good enough for the particular problem situation even if it may be inadequate for universal use. The solution shelf removes any ambiguity since it captures all identified recipes and constraints. If further analysis of the problem situation reveals additional interacting concerns, the localized ontologies can be further extended and refined.

3.15 'BOOTSTRAPPING' EXTERNAL SHELVES

Buchholz [1953] observes that “[a]nother area where the programming facilities of the computer have successfully replaced physical hardware is in the loading of a new program into the computer. There is a load button and a selector switch on

the machine, but they do just barely enough to get the process started. The rest is accomplished by a technique sometimes called the *bootstrap technique*.”

The problem of *Shelf Bootstrapping* will be routinely encountered whenever the OAR framework is applied to a new subject domain.

The initial step of identifying, analyzing and representing recipes for a domain of interest is known as *Shelf Bootstrapping*. This is required because at the outset of domain analysis using OAR, there will be no identified recipes encapsulating intellectual concerns. Over time, the recipe repertory will be populated with concerns belonging to different domains of interest as and when they are analyzed and encapsulated. This activity qualifies as a bootstrap step because it is possible to often commence the omnispersive analysis with recipes available in external shelves that have been populated earlier. Depending on whether a new problem domain is being examined or new situational knowledge in a preexisting problem/subject domain is being analyzed, the shelf bootstrap step may or may not be a prerequisite.

These two scenarios of the shelf bootstrap process are visualized in [Figure 3.10](#).

Depending upon the granularity of the problem under consideration, the composition and inventory of the external shelves varies.

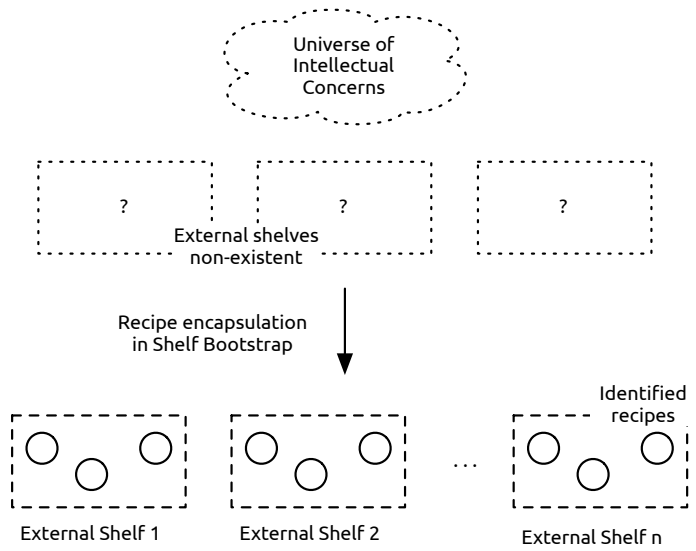
This bootstrap abstraction is a necessary step if the analysis of the problem is started afresh without the benefit of any preexisting relevant recipes. On the other hand, if preexisting repertories with relevant recipes are already available, the OAR process commences with the identification and selection of recipes into a problem domain shelf.

3.16 CONTEXT REFINEMENT

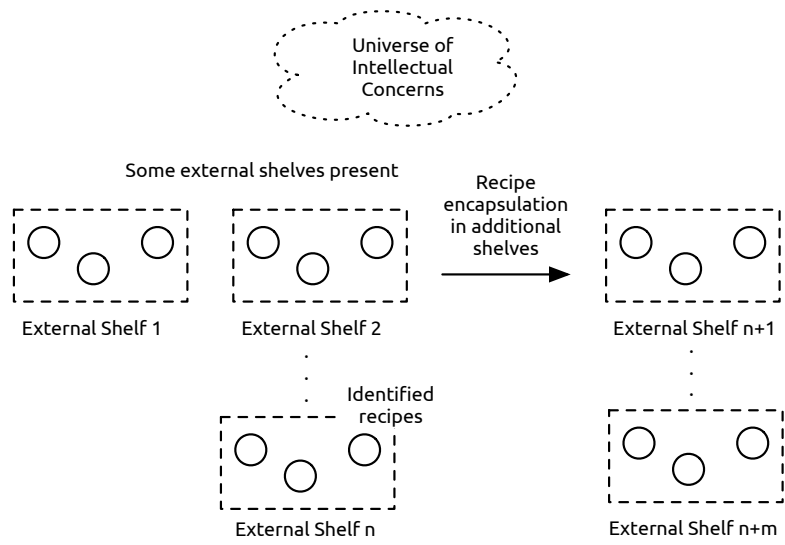
The determination of which recipes are relevant to a problem domain is carried through the process of context refinement. Context specifies the degree of relevance of individual recipes in the problem domain.

As discussed earlier, OAR recipe context (C) is defined as a function of the attributes *Influence* (I) and *Firmness* (F) ([Figure 3.11](#)).

To start with, no *a priori* assumption is made regarding the influence and firmness of the recipes. If analysis suggests the use of a particular recipe, then it is identified as exerting a non-zero influence on the problem under study. In addition, if the recipe falls under the category of an *exemplar* or *best practice* in the discipline, then it is identified as firm. Consequently, the specification composed from the selected recipes will become increasingly *firm* as situational and imposed constraints are satisfied.



(a) No pre-existing external shelves.



(b) Some pre-existing external shelves.

Figure 3.10: Visualization of shelf bootstrap.

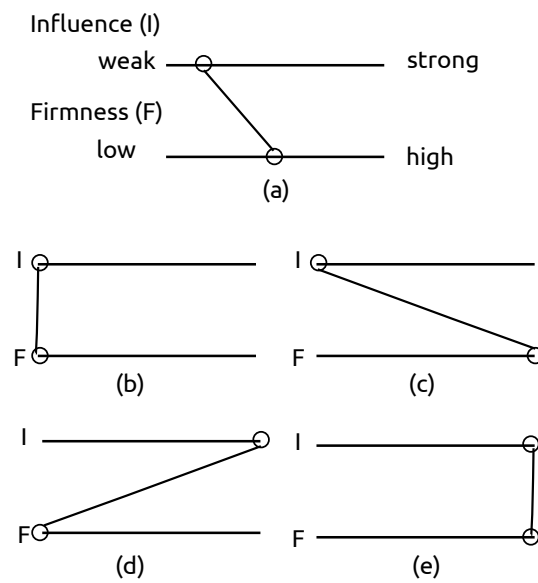


Figure 3.11: Representing context by profiles of the attributes Influence and Firmness.

Though OAR recipe context is a continuous function of two dimensions, in most circumstances, recipe context can be conveniently tagged by discrete labels of the function $C(I, F)$ affecting prototype selection:

1. $C(I = 0, F = 0)$: a context state corresponding to weak influence and low firmness.
2. $C(I = 0, F = 1)$: a context state corresponding to weak influence and high firmness.
3. $C(I = 1, F = 0)$: a context state corresponding to strong influence and low firmness.
4. $C(I = 1, F = 1)$: a context state corresponding to strong influence and high firmness.

These are illustrated in [Figure 3.11b](#) to [Figure 3.11e](#).

The process of context refinement ([Figure 3.12](#)) is carried out by the following steps:

- CTR-1** Identify those recipes that influence the outcome of the process and also those influencing other recipes.
- CTR-2** Determine if there are context connections between the recipes identified in the first step.

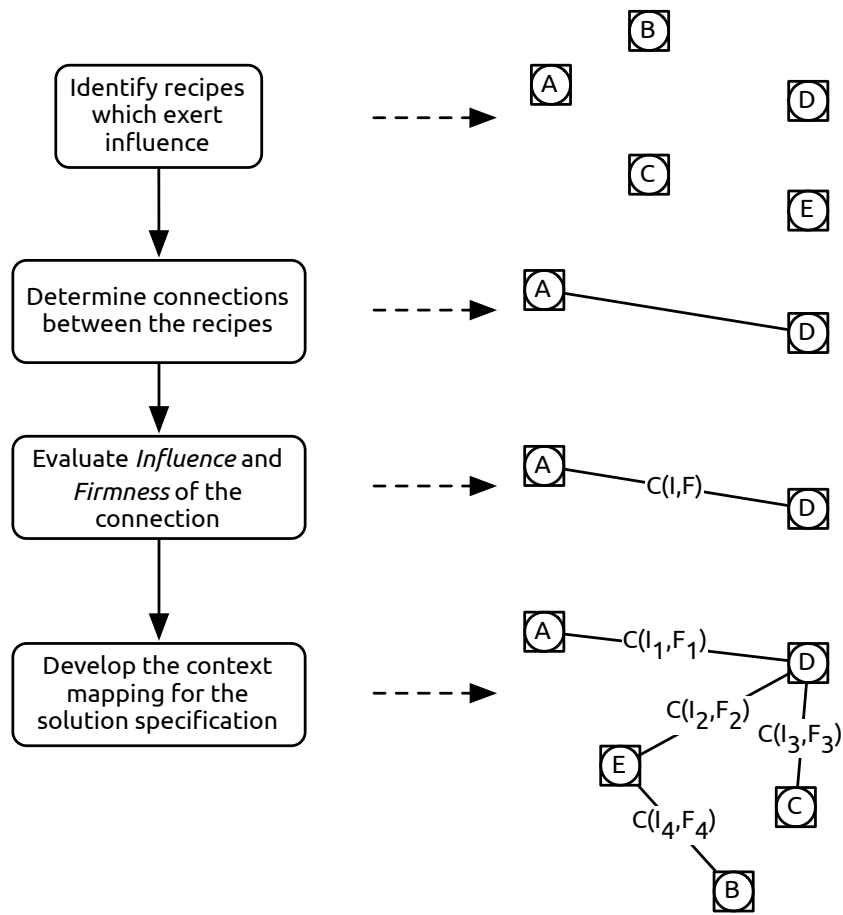


Figure 3.12: The process of context refinement.

CTR-3 Evaluate the Influence and Firmness of the connection and assign a context label to the connection.

CTR-4 Develop the context mapping for the entire specification of the problem.

The first two steps of context refinement in [Figure 3.12](#) may be carried out recursively to obtain a complete mapping of the context relationship in the final specification.

3.17 CONSTRUCTED AND ORGANIC SOLUTION SPECIFICATIONS

The processes of concern refinement and context refinement as described in the previous sections result in a solution specification for the problem under study.

Considering the criteria for a *satisficing* path as “a path that will permit satisfaction at some specified level of all its needs” [Simon, 1956], it can be deduced that in the minimal case, the OAR specification will at least map a satisficing path of connected prototypes. However, if the specification consists entirely of exemplar recipes (archetypes), then the specification can be expected to be optimized.

From this, it can be inferred that when a specification consists only of *archetypes* subject to *firm* constraints, such a specification can only be achieved as a limiting process. In the OAR framework, such a specification is termed as a *constructed specification*. Thus, a constructed specification is a *fully optimized specification*.

On the other hand, if a specification incorporates a *mix* of prototypes and archetypes, or is fully composed only of prototypes, or is also subject to at least one *pliable* constraint, the specification can be identified as an *evolving* or *organic* specification. Such a specification, which is *woven* together, is termed as an *organic specification*.

The presence of organic specifications is a necessary and sufficient condition for handling incomplete prototypes and partial specifications or *specification groups* (which are also similar to overlapping prototype groups). These specifications are *indistinguishable* except for the difference in behavior due to the set of constraints used (i.e., same specifications can exhibit different behavior with different constraints). In a formal sense, these can be considered uniquely different specifications. The idea of specification groups (and prototype and constraint groups) is incorporated for convenience and tractability.

Even though discrete values are assigned to influence and firmness for convenience, it must be noted that in practice the influence and firmness of the contextual relationship between two recipes can vary from very significant to very

insignificant in a continuous manner. Thus fractional values of I and F are real possibilities.

The existence of fractional firmness and fractional influence implies that ‘exact’ solution specifications of the problem are not possible with the available set of recipes in various external shelves and problem domain shelves. Very low values of influence and firmness in context refinement imply that the possible solutions may not be optimal or the understanding of the interacting domains may be incomplete, signifying a need for further analysis of the problem situation. Going by the axiom that “all models are wrong” [Stermann, 2002], i.e., any model is only an approximation, even though fractional values of I and F is the norm in theory, the four discrete context tags are assigned as a working solution. Further work to handle fractional context parameters will help reduce any uncertainty in the resulting OAR specification. However this work is beyond the scope of the present investigation.

3.18 RATIONALE FOR THE STRUCTURE OF OAR

In OAR, an identified concern is abstracted at one or more levels — Concept, Model and Implementation. This is based on the CMI hierarchy of conceptualization proposed earlier in this chapter. Implementation-level is the lowest level (where the concern is put into practice); model level is higher level of abstraction (where representation of the concern is done); concept level is the highest level of abstraction (where understanding of the concern is involved). Labeling the level of abstraction as higher or lower is relative to the circumstances of the problem situation as explained in [Chapter 2](#) in connection with delineating the details of a concern ([Figure 2.2](#)).

According to Warfield [1994], a design of a system has to be as simple as possible in order to reduce situational complexity and cognitive burden. A design will be simplest when it is conceived in terms of ideas (thought processes) and operates with ideas [Warfield, 1994]. This is so because the operations that can be carried out with ideas are limited to only five — generating, clarifying, structuring, interpreting (the structure) and amending the ideas.

In managing the concerns in a workflow, the main ideas employed in the OAR framework are: omnispersion, analysis, reasoning, abstraction, and providing epistemic hooks through concern refinement and context refinement. The manner in which these ideas are used to realize the various facets of the OAR framework is shown in [Table 3.1](#).

Ideas used in OAR	Operations with Ideas	Related features of OAR
Omnispection; Analysis; Reasoning; Abstraction	Generating; Clarifying; Amending	Abstraction of ukes at three levels. Management of Recipes by Shelves.
Reasoning; Concern Refinement	Clarifying; Structuring; Amending	Managing Problem Domain Shelf. Workflow specification.
Reasoning; Con- text refinement	Interpreting structured ideas; Amending	Managing Solution Shelf. Solution Specification

Table 3.1: Ideas in the OAR framework.

The concept of amending ideas can be seen as included in the operation of the OODA loop. This is the process by which abstraction of concerns is carried out to be consistent with the scientific basis of the problem domain when the basis ‘evolves’ or is modified due to new findings in the field.

The hierarchy of the three levels at which ukes can be abstracted, as seen from [Table 3.1](#), is additionally related to the concept of operations with ideas.

OAR is designed to be modular, similar to the modular organization of the DoSM. The management of recipes in shelves provides such modularity. The (identified) concerns from different domains of knowledge that are considered relevant to the problem under study (as revealed by the process of omnispection, analysis and reasoning) are abstracted as recipes (ukes and groups of ukes). These are added to the external shelves in a modular fashion. Organization in shelves encourages the inclusion of all the information that is considered to be relevant to the investigation. This enables use of localized ontologies if required.

The management of recipes using the concept of shelves, and the processes of concern and context refinement form an essential part of the methodology of the OAR framework. The process of concern refinement in tandem with context refinement provides (extensive) epistemological support to the working of OAR — in identifying archetypes and constraints and in the formulation of the workflow and the solution that is obtained therefrom. They provide the means for formulating the workflow, i.e., modeling the workflow in the problem domain shelf and obtaining the solution specification in the solution shelf. The solution shelf may result in a solution that may be directly implemented, or it may require further translation by invoking a suitable translation engine. As shown in [Table 3.2](#),

the kind of translation engine depends on the nature of the study and the desired solution.

Kind of Solution	Kind of Translation Engine
Quantitative evaluation of computed data	Computational algorithms incorporated in software programs
Operation of devices as per the solution specification	Interface for actuating the device and related control programs
Decision-making process involving judgement	Human agent making decision based on the solution specification

Table 3.2: Some OAR translation engines.

The data handling and implementation details as necessary for these translation engines themselves could be abstracted as suitable recipes in external shelves. As pointed out in [Section 3.11](#), support for the intellectual burden of dealing with implementation details is provided by encapsulating them as recipes at a suitable level of abstraction; depending on the granularity of the problem, these may involve high or low-level concerns, as explained in [Section 2.2.1](#).

3.19 NATURE OF THE OAR FRAMEWORK

The OAR framework is formulated as an epistemic framework for managing the intellectual concerns in scientific workflows. A framework is a generic term representing a scaffolding in which a number of entities are arranged in order to realize a certain functionality. A variety of frameworks are used in scientific and software engineering domains.

A theoretical framework is a collection of interrelated concepts and methods developed for analyzing and solving a problem. In the OAR framework, a set of logical, conceptual and procedural principles are introduced for the management of scientific workflows. Hence, OAR may be termed a Theoretical Framework. It is epistemic because of the emphasis on capturing the intellectual concerns in the problem under study and deliberately encouraging their verification with reference to the scientific basis of the problem.

The design of OAR is modular, like DoSM, consisting of Foundation, Theory and Methodology. As explained in the previous section, the structure and working of OAR incorporate the evolving nature of science, hierarchy of conceptualization, omnisppection, and the logical processes of analysis, reasoning and abstraction. These form the Foundation and Theory of OAR. Abstracting concerns in terms of ukes and groups of ukes (recipes), use of context to identify relation between recipes, the management of recipes in shelves and the processes of concern refinement and context refinement constitute the Methodology.

While the Applications block is conspicuous by its absence in the current formulation of the OAR framework, the three applications in chapters 4 to 6 illustrate the use and applicability of OAR to workflow management and provide a first step in this direction.

3.20 SUMMARY AND CONCLUSION

The foundation, theory and methodology of Omnispective Analysis and Reasoning have been presented in this chapter. OAR is conceived and developed as an epistemic framework for the management — particularly intellectual concern management — of scientific workflows. Following the reasoning and conjecture presented in the [Chapter 2](#), additional considerations are presented to support the necessity of developing an enhanced definition of scientific workflow and of lifting focus from low-level implementation details in order to effectively manage intellectual concerns.

An enhanced definition of scientific workflow is formulated in Definition 1 as: *A scientific workflow is a representation of any logical, systematic and repeatable inquiry, investigation and corresponding set of actions.*

In addition to incorporating the traditional view of scientific workflows, this definition also extends the scope of ordered analysis and investigation to any generic problem scenario.

A hierarchy of conceptualization of an entity at concept, model and implementation levels is postulated.

The evolving nature of science, the method of science for fixing belief and the limitations of human comprehension as enunciated in the law of triadic compatibility are utilized in the conceptualization and formulation of the framework.

The OAR process for formulating the workflow for a problem starts with omnisppection and the processes of abstracting the concerns in the problem domain as ukes and groups of ukes (recipes) at three levels of abstraction — concept, model and implementation levels.

Context of a recipe is defined in terms of the well-formedness (firmness) of the recipe and its influence on other recipes.

The workflow formulation and implementation is carried out by management of recipes in shelves as unordered collections of recipes. The process of concern refinement is utilized in order to formulate the problem-domain shelf. An unambiguous solution specification is realized from the problem-domain shelf using context refinement. These OAR processes enable focus on intellectual concerns, while facilitating localized ontologies pertinent to the particular instance of problem formulation. The abstraction and contextualization of recipes further enables the identification and enunciation of concerns that may not be immediately apparent, but are relevant to the problem.

The description of OAR is followed by an elaboration of the rationale of its structure and design, along with a discussion of the nature of the framework. OAR is a modular framework, consisting of a Foundation, Theory and Methodology. The Foundation and Theory of OAR are based on the evolving nature of science, the CMI hierarchy, omnispersion, and the logical processes of analysis, reasoning and abstraction. Methodology consists of recipe formulation, use of context to identify relation between recipes, and management of recipes in shelves by concern and context refinement.

In the next part of the thesis, the use of the OAR framework is illustrated by three different applications.

Part III

Proof of Concept

DEMONSTRATING A NEW FRAMEWORK

PROOF OF CONCEPT, CASE STUDY AND ILLUSTRATION BY APPLICATION

The scientific method requires that a set of new ideas and methods need to be demonstrated and evaluated before they can be applied to general practice. This demonstration and evaluation can be achieved in a number of ways — as a Proof of Concept (PoC), a Case Study or Illustration by Application (IbA). It may be pointed out that the terms PoC and Case Study are frequently used without clear distinction of their scope and purpose and are sometimes confused with PoC for seeking commercial support for new ventures or with Case Study Research in fields like Sociology. Semantics apart, the exercise of demonstrating the potential and applicability of new concepts and methods is an important step in integrating the new-found knowledge into the scientific stream of the relevant research domain.

Proof of Concept, as generally understood, consists of a made-up example, created for the sole purpose of illustrating the ideas, concepts, frameworks and methods. The example may be small or incomplete and is primarily intended to show the consistency of the ideas involved, but need not necessarily show the manner of applying them in practice nor their desirability or usefulness.

Hartmanis [1995] contends that in the domain of Computer Science, after a new idea or method is conceptualized, a demonstration provides adequate proof of concept. This is attributed to the nature of computer science which is different from natural sciences and deals with concepts and ideas rather than physical entities. According to Tichy [1998], demonstration of an idea or method provides a proof of concept in the “engineering sense” and shows that it has potential for use and is suitable for further extensive development.

A fully developed case study, on the other hand, consists of a detailed and in-depth application of the proposed ideas, methods and frameworks, through all the stages of a real problem situation. This has the advantage of thoroughly evaluating and testing the fitness and applicability and understanding the benefits and limitations. However, this would be feasible only if the proponents of the new ideas are themselves part of a team tackling a new research problem wherein they might attempt to apply and evaluate the new ideas, while at the same time trying

to address the research problem. This becomes even more difficult if the research is an ongoing one, since the risk-averse considerations of project management do not normally encourage the use and adoption of new and untried ideas in an existing project. Further, applying new ideas that emanated from 'outsiders' (who are not part of the core team) is usually considered as adding project risk and uncertainty. Real projects are driven by deliverable outcomes and not by the need to evaluate new ideas, unless evaluating the new ideas itself is the goal of the project. A similar sentiment is expressed by Flint [2006] to justify the choice of the PoC approach.

A middle ground between the PoC and Case Study approaches is Illustration by Application (IbA). Here one applies the new ideas and methods to a part or whole of an existing or new problem scenario without necessarily being part of a team working on the particular problem. Certain areas and characteristics of the problem are used to illustrate and bring out the applicability and potential usefulness of the new ideas and concepts. Although IbA is not as exhaustive and complete as a Case Study, it is much more concrete in demonstrating applicability and merit than a simple PoC involving an entirely made-up example.

In this thesis the OAR framework is demonstrated following Illustration by Application. Three different examples are considered. The primary aim is to show clearly the working of the different processes of the OAR framework; the focus is not per se on the novelty or usefulness of the outcomes of the illustrations employed. Here a parallel can be drawn with the demonstration of scientific concepts using simple and well-articulated experiments.

The first application presented is a workflow for folding an Iris Flower, a problem in the domain of origami. It is a known problem with logical and clearly defined steps and is elaborate enough to demonstrate the features of the OAR framework.

The second application, contextual course design, is a new problem situation in the educational domain and illustrates the applicability of OAR to a generic workflow across disciplines.

The third illustration is application of OAR to the analysis of a complex system and demonstrates the potential of OAR for the analysis, understanding and management of such systems.

These three applications, which demonstrate the applicability and potential of the OAR framework and have appeared in refereed publications [Chemboli and Boughton, 2011, 2012a,b] are discussed in the following three chapters.

4

ORIGAMI FOLDING WORKFLOW

Before you can think outside the box, you have to be in the box.

Tharp and Reiter [2006]

4.1	Paper Folding as a Scientific Workflow	82
4.2	Folding the Iris Flower	82
4.3	Applying OAR to the Iris Flower Workflow	85
4.3.1	Identifying Relevant Archetypes and Constraints	86
4.3.2	Formulating the Solution Specification	86
4.4	Implementing the Solution Specification	87
4.5	Summary and Conclusion	88

The Omnispective Analysis and Reasoning framework has been presented in the previous chapter. Application of the OAR framework is demonstrated in this chapter by considering a *scientific workflow problem* in origami — the process of folding paper into shapes representing various objects.

This chapter is organized as follows. Firstly, the characteristics of origami that qualify it as a scientific workflow are presented. Secondly, the workflow for folding an *Iris Flower* is presented in the fashion of the current practices of scientific workflow management, in which a workflow is considered as a sequence of “tasks and their dependencies.” The shortcomings in intellectual concern management in this approach are highlighted. Next, the Iris Flower workflow is considered again using the OAR framework. In particular, this demonstration illustrates how the OAR framework facilitates the mapping of an implementation-level workflow (for folding the Iris Flower) to intellectual concerns at the *Concept* and *Model* levels (origami intellectual concerns). The processes of shelf representation, concern and context refinement to obtain the solution specification and its corresponding implementation are demonstrated.

4.1 PAPER FOLDING AS A SCIENTIFIC WORKFLOW

The art of paper folding holds great fascination for artisans as well as scientists and engineers [Demaine and O'Rourke, 2007]. The concepts of origami find wide applications in many technological innovations like arterial stents, solar panels for spaceships, novel and innovative architectural designs and mathematical algorithms for systematic folding of varied and intricate shapes from a sheet [Wang-Iverson, Lang, and Yim, 2011].

Origami folding embodies several of the characteristics of scientific practice. Starting with the selection of suitable paper, techniques for handling the paper, folds, bases and the sequence of folding operations are all well-defined and have been established by exploration and experimentation by origami practitioners. These constitute the intellectual concerns in origami and can be abstracted as recipes for the purpose of formulating an origami pattern.

The folding operations are unambiguously defined sequences of steps and have a clear dependence on the previously executed steps. Thus the processes of origami constitute logical, systematic and structured activities and display the features of contextual relations, interactions and constraints at play.

An origami workflow not only possesses the basic characteristics of a scientific workflow, but also is sufficiently adequate to provide a clear demonstration of the salient features of the OAR framework.

4.2 FOLDING THE IRIS FLOWER

The iris flower is a traditional figure in origami construction [Kenneway, 1987]. It is generally constructed starting either with a *preliminary base* or with a *frog base*. Figure 4.1 illustrates the workflow modules and steps for folding the iris flower.

Starting with a Frog Base, a sequence of structured steps is carried out. The workflow consists of a number of sub-workflows or modules which are composed of additional steps — this reflects the current status of scientific workflow practice. As discussed in chapters 2 and 3, the current practice of scientific workflows provides features to design and orchestrate experimental and computational steps in data collection, organization and analysis. This allows effective handling of research and supports automation of repetitive and computationally intensive tasks. However, the specification of workflows has more focus on low-level details of data variables, process branching, looping and control and other chores connected with system-level tasks of control and management.

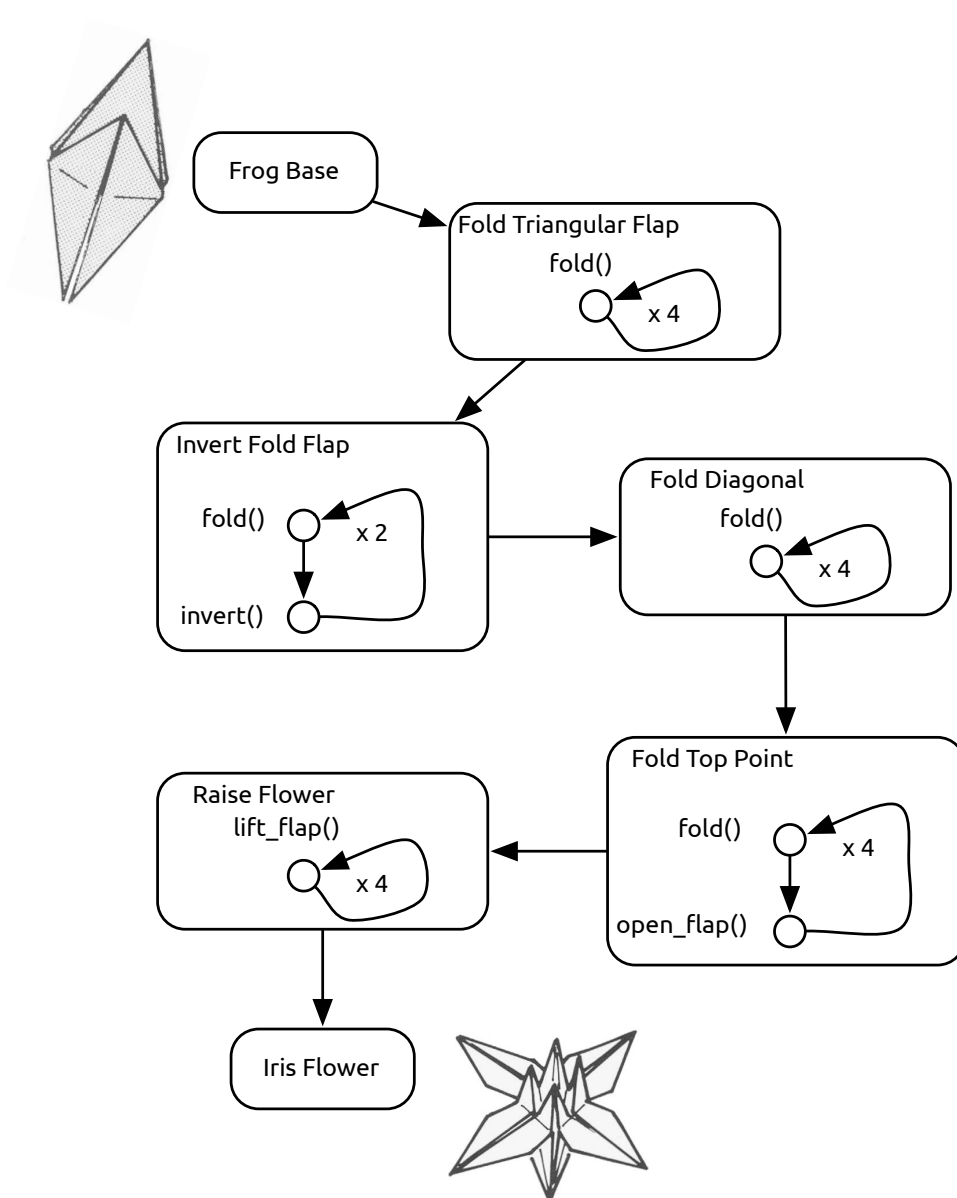


Figure 4.1: Workflow for folding the Iris flower starting with a Frog Base.

The workflow illustrated in [Figure 4.1](#) does not have any explicit provision for identifying, verifying and validating intellectual concerns of the problem domain. Also, no deliberate focus exists on ensuring and verifying that the correct scientific workflow is modeled and implemented for a given problem situation. Any validation done is performed outside the workflow as an independent activity.

Based on these considerations, the origami workflow described is an implementation–level workflow. This is depicted in [Figure 4.2](#)

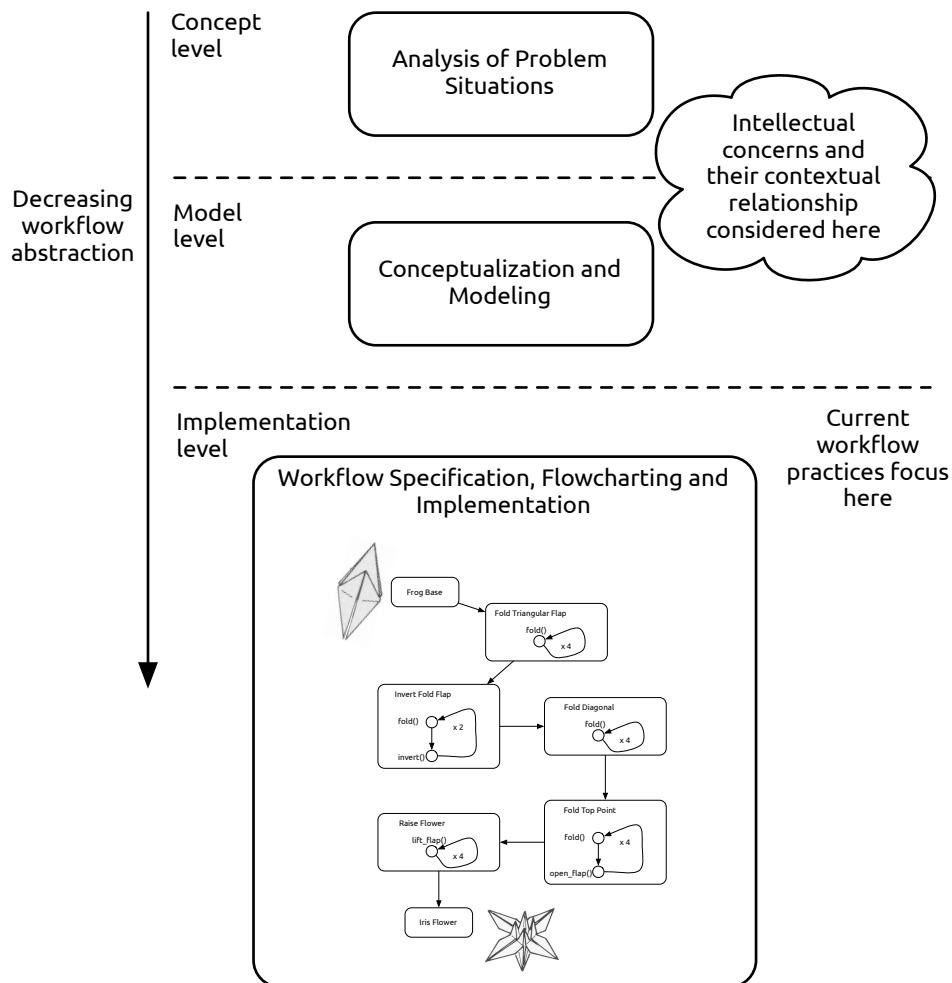


Figure 4.2: Illustrating the implementation–level focus of current scientific workflow practice.

In light of the discussion in chapters 2 and 3, it can be discerned from [Figure 4.2](#) that, by itself, an implementation–level workflow does not go into the underlying principles nor does it provide end–to–end correlation between the science, modeling and implementation. Such workflow takes away focus from the rationale and intent of the scientific activity and obscures the interpretation and analysis of the results in terms of the underlying science.

4.3 APPLYING OAR TO THE IRIS FLOWER WORKFLOW

The origami Iris Flower workflow is revisited in this section by using the OAR framework. The OAR process starts with examining the problem situation and identifying the (intellectual) concerns and their interactions.

Folds, bases, paper characteristics, folding algorithms and steps, folding techniques and types, visual notations for folding sequences etc. are the various concerns involved in origami. As explained in [Chapter 3](#), the idea of concept-level, model-level and implementation-level concerns is a theoretical foundation for the OAR framework. Every recipe within the framework can be evaluated at any of the three levels depending upon the context of the problem analysis. Separation of recipes into concept, model and implementation levels does not occur until the recipe is considered for analysis. Depending on the outcome of the analysis, and the relevance of the recipe to the problem situation under study, the recipe is considered at an appropriate level of the CMI hierarchy.

The recipes are managed in a number of external domain shelves corresponding to the folding vocabulary and procedures. Depending upon the granularity of the problem under consideration, the composition and inventory of the external shelves varies. For example, if the problem scenario is a comparison of different art forms (such as origami, painting, sculpting, etc.), one may consider all the known recipes pertaining to origami as the contents of a contiguous origami shelf. On the other hand, for a problem situation within origami, such as folding of the Iris Flower, the relevant recipes pertaining to origami may be viewed as members of different external shelves that capture the (intellectual) concerns in origami.

External shelves with recipes that can be identified as relevant for formulating a specification for the iris flower workflow are given in [Figure 4.3](#). This corresponds to the bootstrapping process when starting an origami problem as a new workflow, with no preexisting recipes in the external shelves.

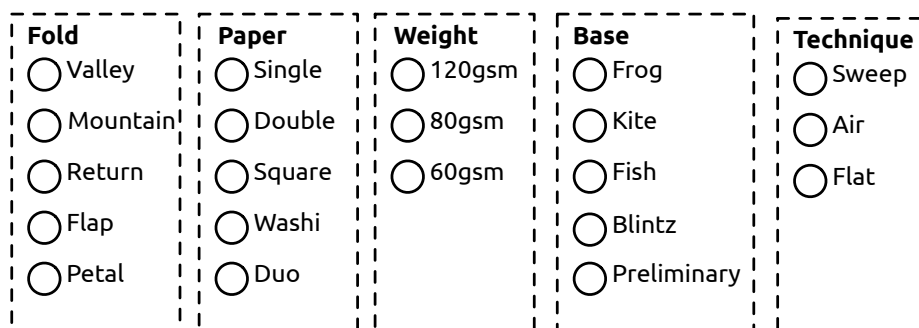


Figure 4.3: External shelves and prototypes.

The list of recipes illustrated here is not exhaustive; but is sufficient to develop an OAR specification for the folding of an iris flower.

4.3.1 Identifying Relevant Archetypes and Constraints

The process of concern refinement is now carried out to select the specific recipes that can be imported into the problem domain shelf. The Iris Flower can be folded starting either from the *Preliminary Base* or the *Frog Base*. The recipes for the Preliminary Base and Frog Base are selected from the Base external shelf.

The *Flat* recipe from the *Technique* external shelf is identified as a constraint. Though the Iris Flower can be constructed using either the Flat or the Air technique of folding, the Flat technique is chosen here as a personal choice of the practitioner, which now acts as a constraint. It is easier to fold the Iris Flower if a paper of a relatively lighter weight is used, and accordingly the *60 gsm* prototype is selected into the Problem–domain shelf.

The resultant problem–domain shelf is illustrated in [Figure 4.4](#).

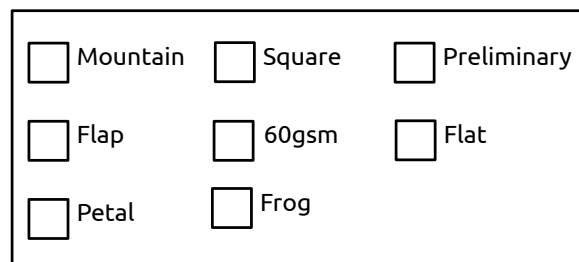


Figure 4.4: Selected prototypes in the Problem–domain shelf for the Iris Flower specification.

4.3.2 Formulating the Solution Specification

The process of context refinement is carried out to define the solution specification for the Iris Flower as follows.

ss-1 The Frog base archetype is implemented using the Flat technique with a Square paper. Therefore, this is an archetype that exerts a high degree of influence on the workflow.

Flat C(I = 1, F = 1) Square

Square C(I = 1, F = 1) Frog

ss-2 Although the Preliminary Base can also be used as a starting point for the Iris Flower, it is not as convenient as starting with the Frog Base.

Preliminary C(I = 0, F = 1) Iris

Frog C(I = 1, F = 1) Iris

ss-3 The petals of the Iris Flower can be formed by folding the Frog Base further along the flaps. Thus the process to form the petal from the Frog Base is firmly defined. This is enumerated in the following context relation:

Frog C(I = 1, F = 1) Petal

ss-4 Four symmetric petals are formed in order to construct the Iris Flower. Not only is this a well-articulated process, but is also a necessary step. Hence:

Petal C(I = 1, F = 1) Iris

With these constraints, the following solution specification, illustrated in [Figure 4.5](#) is obtained.

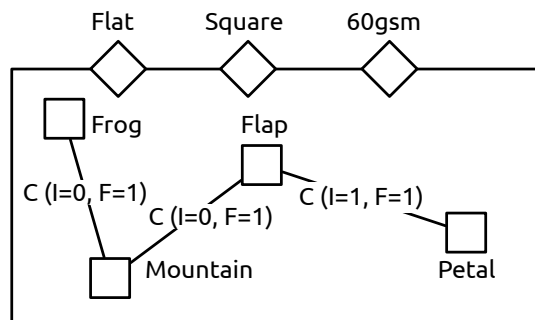


Figure 4.5: Origami Iris Flower specification.

It may be noted that the recipes imported into the Problem Domain Shelf and utilized in the Solution Shelf for specifying the solution, are firm; they are either archetypes or constraints. Accordingly, as explained in [Section 3.17](#), the resultant solution specification is a *constructed specification* and may be considered as an optimized specification.

4.4 IMPLEMENTING THE SOLUTION SPECIFICATION

Recipe membership in an external shelf is not required to be exclusive. Depending on the granularity of the problem analysis, the same recipe (perhaps a solution shelf formulated earlier) may be considered across multiple external shelves. In addition, multiple instances of solution specifications for a problem situation may

also form a family of recipes in related shelves. Although there is no requirement of uniqueness for either shelf categorization or recipe membership, individual recipes are still unambiguously specified and translated even though they may correspond to different pathways of translation depending on the context of the problem situation.

The solution shelf in [Figure 4.5](#) is a model-level instance of solution formulation for the problem of folding the Iris Flower, which is consistent with recipes at the concept-level and represents the rationale of the steps in the workflow. The actual task of folding the iris flower is performed by an implementation-level workflow, which is obtained by repeating the OAR process at a lower level of abstraction.

To implement the solution specification of [Figure 4.5](#), it is now considered as a recipe in an external shelf of origami flower folding specifications. A new OAR process is now initiated by carrying out concern refinement and context refinement for the new workflow. The process is illustrated in [Figure 4.6](#).

In the new OAR process now initiated, recipes of the Iris Flower folding specification, the Traditional School of origami folding and a Translation Engine recipe are selected into the problem-domain shelf. The solution shelf obtained represents the workflow at the implementation-level. Depending on the choice of the Translation Engine recipe, this is equivalent to the implementation-level workflow shown in [Figure 4.1](#).

4.5 SUMMARY AND CONCLUSION

The application of the OAR framework has been illustrated in this chapter through the formulation of a workflow for origami. It is seen that an origami workflow is a structured activity with interdependent tasks and exhibits typical characteristics of scientific workflows.

In the first instance a workflow for folding the Iris flower is given, following the current approach of scientific workflow management. As shown in [Figure 4.2](#), this workflow is an implementation-level workflow, which does not map the intellectual concerns of origami (folds, paper types, techniques etc.) and the rationale and intent of the steps of workflow implementation.

Next, the OAR framework is applied to the origami Iris Flower workflow. After examining the problem situation (folding an origami shape) from different perspectives and domains relevant to origami (for e.g., types of folds, paper types, origami techniques etc.), relevant intellectual concerns are abstracted, categorized and enumerated in an unambiguous and reusable manner, as recipes in external shelves. Following this, the processes of concern refinement and context

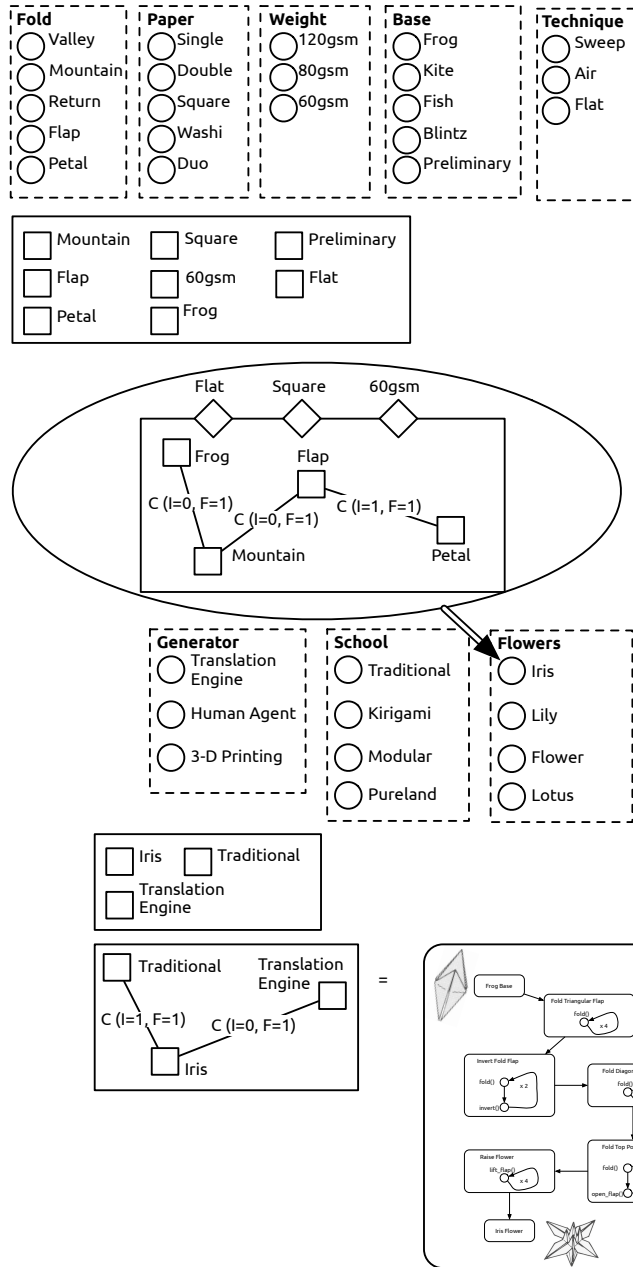


Figure 4.6: Formulating a solution specification for implementing the Iris Flower solution shelf of Figure 4.5.

refinement are performed to formulate a model-level solution specification for the Iris Flower folding workflow in terms of and conforming to the selected recipes. This solution-specification recipe is then used for further developing an implementation-level solution shelf, which corresponds to the step-by-step workflow available for the origami task.

This application of the OAR framework:

1. Illustrates the lifting of focus of the origami task from low-level folding concerns, facilitating their analysis in terms of the concepts of origami such as fold techniques, choice of paper types and choosing a particular path of folding operations.
2. Demonstrates the basis for validating an implementation-level workflow (the sequence of folding steps) against the tradeoffs that can be evaluated for the model and concept-level concerns.
3. Shows shelf management in action, which enables the formulation of a problem solution in terms of preexisting recipes that capture concerns from domains of influence.
4. Provides an illustration of the OAR processes of concern refinement and context refinement in action.
5. Presents a comprehensive and yet simple enough example of the application of OAR to the abstraction, analysis, formulation and orchestration of a scientific workflow at different levels of granularity.

It can be seen that while the second solution specification of [Figure 4.6](#) provides an implementation workflow for folding an Iris Flower shape starting from the initial concepts and ideas of origami, it is by no means the only possible implementation available. Depending on the recipes selected from the Origami School external shelf and the Generator external shelf, the solution specification may perhaps represent a physical object of a constructed iris flower instead of the implementation-level workflow specified in this chapter. Thus, multiple pathways of translation exist depending on the context of the problem situation, with each solution shelf as an unambiguous, but yet related recipe. The relevant solution is chosen depending on whether the desired outcome is a description of the sequence of 'folds' for constructing the Iris flower or an Iris flower artifact (folded from paper).

5

CONTEXTUALIZING COURSE DESIGN

[I]t is helpful to remember that what the student does is actually more important in determining what is learned than what the teacher does.

Shuell [1986]

5.1	Learning, Teaching and Course Design	92
5.1.1	Learning Outcomes	94
5.1.2	Translating Learning Outcomes to Course Design	95
5.2	Contextualizing Course Design	96
5.3	Translating Learning Outcomes for COMP8120	97
5.3.1	Learning Outcomes for COMP8120	97
5.3.2	Analyzing Context for LO-1 and LO-2	98
5.3.3	Analyzing Context for LO-3	98
5.3.4	Analyzing Context for LO-4	99
5.3.5	Analyzing Context for LO-5	99
5.4	Solution Specification for LO-5	99
5.4.1	Initializing External Shelves	99
5.4.2	Identifying Relevant Archetypes and Constraints	100
5.4.3	Solution Shelf for LO-5	101
5.4.4	Implementing the Solution Specification	102
5.5	Summary and Conclusion	102

The OAR framework can be applied for the analysis, study and management of any generic problem situation that conforms to the basic nature of a scientific activity, i.e., the problem situation can be represented and analyzed in terms of logical and repeatable steps and procedures. This follows from the enhanced definition of a scientific workflow presented earlier in [Chapter 3](#), and also from the epistemic nature of omnisppection, analysis and reasoning.

In this chapter, the demonstration of the applicability of the OAR framework in tackling scientific problems across multiple disciplines is taken forward by presenting its application for contextualizing educational course design. The lack of effective means to capture the rationale for design decisions makes it

difficult to validate the design of a course against learning outcomes. The OAR framework facilitates the capture of intellectual concerns and mapping them to the formulation and implementation of a workflow. Hence it can be applied to develop a process for mapping learning outcomes to course design.

This chapter is organized as follows. The problem of translating learning outcomes to the design and development of a course is considered. This is followed by the OAR process for contextualizing course design. This application of the OAR framework is illustrated by considering the process of translating learning outcomes to the design of a course using the Moodle Learning Management System (LMS).

5.1 LEARNING, TEACHING AND COURSE DESIGN

Educational systems are complex systems and operate in a context which keeps on changing with changing societal, industrial and economic activities. The goals of education span a very wide range — from providing simple functional literacy to gaining competence in technology and management to attaining deep knowledge and scholarship -- depending on the motivation and the available incentives and choices to the learners and teachers.

Theoretical and practical research in the field of teaching and learning covers a number of domains like psychology, sociology, physical sciences, engineering and information technology. The report on Scientific Research in Education [National Research Council, 2002] gives a high-level review of the “science and practice of scientific educational research.” The report emphasizes that all current and future research in the educational field should “account for the context of the research, align with a conceptual framework, reflect careful and thorough reasoning and disclose results to encourage debate in the scientific community.”

In the multifaceted arena of education, learning and teaching are two important activities for which effectiveness depends to a considerable extent on the design and delivery of the curriculum. The theory of constructive alignment [Biggs and Tang, 2007] considers how to design a curriculum with optimum use of all the teaching and assessment tools to realize a good teaching environment. A well-designed curriculum in which the learning objectives are mapped to the learning outcomes together with assessment aligned to the aims of teaching will produce conditions for quality learning [Csapo, 2009].

In general a curriculum is a description that specifies, for given subject matter to be taught, the course content, learning objectives and expected outcomes for the course. An outline of content, method of teaching, method of assessing student

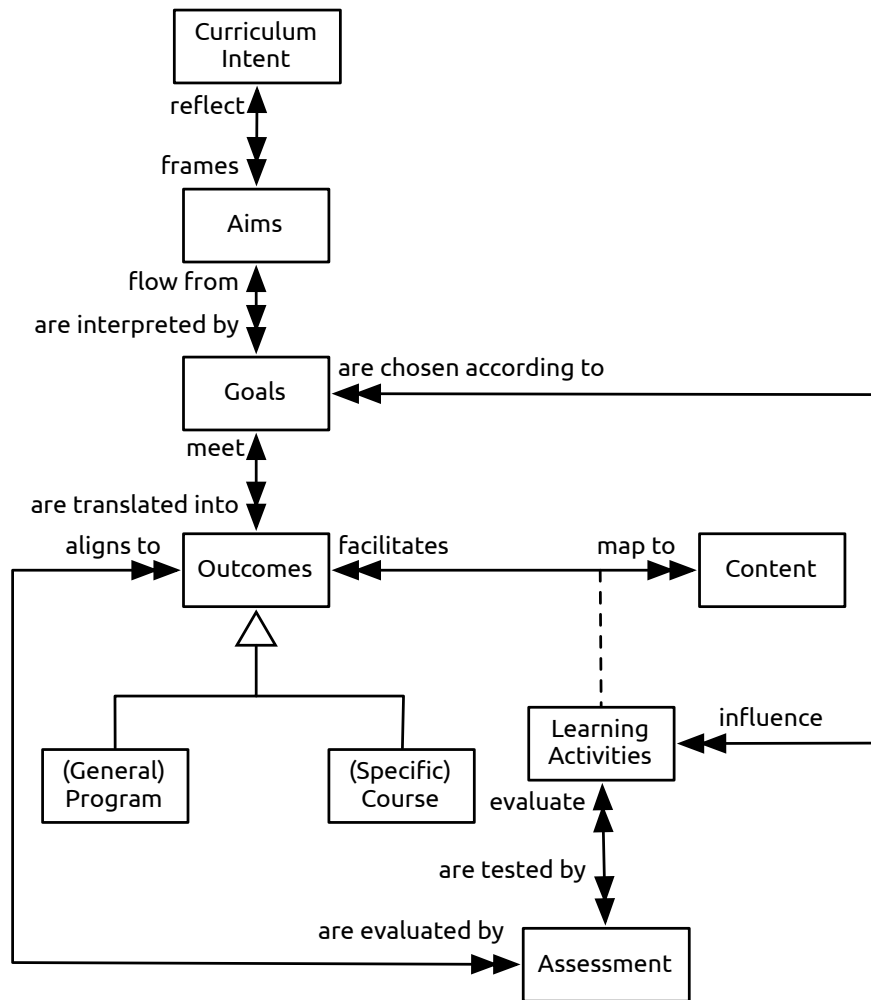


Figure 5.1: A model of curriculum design. Copyright ©2010 Johns-Boast, L. Adapted with permission.

learning and student participation is also included. It is also desirable to include provision for taking care of students of different levels of learning capability and inclination and also to allow for the preferences and operating/teaching practices employed by the teacher delivering the course. In addition, it is also important to properly plan the interactions between teachers and students and include them in the curriculum.

The way the learning, teaching and assessment approaches are organized and practiced is usually represented as a curriculum model. A variety of models for curriculum design [Marsh, 2009; Walker, 1971] have been proposed and used over time. A brief overview of a number of curriculum models is given by O'Neill [2010]. An in-depth exposition of issues in curriculum development can be found in the scholarly work of Biggs and Tang [2007]. A curriculum design model given by Johns-Boast [June 2010, private communication] is utilized in this chapter. This is illustrated in [Figure 5.1](#).

In this model, the aims of the curriculum are decided based on the primary intent and purpose of the course. These aims are represented in terms of the goals which in turn are translated into the course outcomes. The course content is designed and organized to be of adequate depth and breadth to realize these outcomes. Learning activities are planned in accordance with the desired outcomes. The learning activities and course outcomes are tested and evaluated by suitable assessment procedures.

5.1.1 Learning Outcomes

Teaching a course requires the planning of learning outcomes and goals. These are then mapped to existing educational technologies to deliver the course. However, current learning technologies often require significant effort to design and organize courses. As a result, teachers are forced to concentrate their efforts on the quirks and methods of the underlying technological platform which can take their focus away from the goals and outcomes of the course. The activities of online assessment and evaluation tend to become more of a chore, rather than means to support the goals and objectives of a course. This also makes it difficult to share ideas and educational objects across courses and adapt them in a reusable manner.

These difficulties may be overcome by formalizing the implicit rationale in learning outcomes by identifying and managing the intellectual concerns associated with the design and implementation of a course and explicitly capturing the associated learning context. In terms of the OAR framework, this corresponds

to abstracting the learning outcomes as recipes and contextualizing them with reference to course design and delivery.

5.1.2 Translating Learning Outcomes to Course Design

It is recognized that the application of theory of constructive alignment [Biggs and Tang, 2007] to design course content results in an effective approach to course delivery. Constructive alignment requires that the teacher align the *planned learning activities* with the *learning outcomes* [Houghton, 2004]. However, because of the different natures of the domains of course design and learning outcomes and the Learning Management System, one often encounters mismatch and difficulty in mapping the learning outcomes to the *resources* and *processes* of the LMS. For instance, while learning outcomes for a course may be formulated in terms of the desired goals and means for assessment and evaluation of effective learning, the implementation of the course in an LMS is undertaken in terms of LMS resources such as lessons and exercise blocks, and widgets like buttons and checkboxes, accompanied by participatory student activities such as discussion forums, quizzes and chats. Therefore there is a need to provide a link between the pedagogical concepts and their implementation in an LMS. Online instruments would further help this linking of LMS activities and resources to learning outcomes.

The Moodle LMS [Moodle Community, 2012] is used in the course design discussed in this chapter. Moodle is a web application that can be used as a tool for creating a Virtual Learning Environment (VLE) consisting of online websites for teaching and learning. It is a free application available under the GNU Public License. A good structural overview of Moodle is presented by Drechsler [2011].

A Moodle course is structured in the form of *pages* containing *Activities*, *Resources* and *Blocks*. Activities allow students to interact amongst themselves and with the teacher. Resources consist of items that are added by the teacher to support learning. Blocks provide additional information and links to additional resources and reading materials. All these are provided by the course designer or teacher.

Some salient features in the use of Moodle are:

1. Small groups to very large groups of students can be addressed.
2. It can be used to organize fully online courses or blended courses that involve direct engagement in teaching supported and supplemented by online interaction.

3. The activity modules like forums, chats and wikis can be used to build collaborative learning communities for delivery of course content combined with assessment of the learning outcomes through assignments, quizzes and online chats.

The following difficulties are encountered while translating the intent and learning outcomes of a course into activities and resources in an LMS implementation:

1. Resolving the differences between language structure and terminology used in defining learning outcomes and the LMS implementation.
2. There is a disconnect between stating the goals of a course and developing a course in the LMS.
3. Lack of an effective means to capture the *rationale* for design decisions in the LMS (for instance, which activity/resource is chosen and why in Moodle, what are the tradeoffs in using different resources to achieve the same outcome).
4. Communication and collaboration issues that emerge in the interaction between course designers, lecturers and LMS developers.

5.2 CONTEXTUALIZING COURSE DESIGN

In order to analyze and organize the intended learning, teaching and assessment processes, the OAR framework is employed in the following manner. Firstly, learning objectives and outcomes of a course are captured in accord with models of curriculum design. Secondly, these *concept level* concerns are then expressed as *model level* concerns in the form of a specification in terms of available educational models. Finally, the design decisions of a course are implemented using the LMS and associated educational technologies, in order to enable a two-way mapping between the learning outcomes, and the resources in the underlying technological platform. This will result in two benefits: mapping of learning outcomes to the underlying educational technology; and facilitation of better evaluation of the effectiveness and suitability of various teaching and learning approaches and outcomes.

The preliminary process of understanding the rationale for learning outcomes begins with identifying relevant *concerns* of interest in the course under consideration. The OAR framework processes of concern and context refinement are then applied to analyze the desired learning outcomes against accepted exemplars

(*external prototypes*) presented by Ramsden [2003] and Biggs and Tang [2007] as applicable within the context of research-based education (*constraint factor*) at the Australian National University (ANU) [Strazdins, 2007].

The following steps are executed after the learning outcomes for a course are determined:

1. Choose a particular outcome for the course.
2. Identify the characteristics of the outcome.
3. Identify the Moodle resource/activity or associated technology which supports the development of these characteristics.
4. Develop the Moodle activity/resource to satisfy the outcome.

5.3 TRANSLATING LEARNING OUTCOMES FOR COMP8120

COMP8120 — System/Software Development Methodologies, was a mandatory course in the Master of Software Engineering Program at The Australian National University. The Master of Software Engineering was aimed at updating and equipping software and systems engineering professionals with a repertoire of best practices and paradigms, empowering them to make informed decisions and evaluate their impact in a measured and systematic manner.

The prerequisites for the program are quite varied, ranging from systems engineering, electronics engineering, software engineering to software intensive systems, biotechnology and policy and governance. Relevant professional experience of two or more years is also necessary for admission to the program. Thus the participants have diverse backgrounds, with their individual styles of learning, interacting and communicating, but not necessarily versed in the aims of the COMP8120 course.

As it was pointed out in [Section 5.1](#), designing flexibility in addressing the concerns related to the aims, outcomes and assessment of the course by way of mapping the goals to the learning outcomes and aligning assessment to the aims and objectives of the course, results in robust curriculum design which is adaptive and responsive to the learning outcomes.

5.3.1 Learning Outcomes for COMP8120

LO-1 The students will develop an appreciation of past, contemporary and emerging software/systems analysis and design techniques.

- LO-2 The students will be able to evaluate and understand the scope, applicability, limitations and extensibility of software/systems analysis and design techniques.
- LO-3 Students will be able to apply and expand the methodologies to accommodate appropriate multidisciplinary knowledge, and inform decisions affecting problem situations.
- LO-4 Students will use the approaches learnt to identify necessary improvements and inform recommendations about their realization using appropriate and well-informed inputs from multiple sources of knowledge and expertise.
- LO-5 The students will reflect, inform and communicate the issues in the integration approach identified for improving a problem situation to the satisfaction of all stakeholders identified.

5.3.2 Analyzing Context for LO-1 and LO-2

It is assumed by the designers of the course that the students may not be conversant with system/software development methodologies (SSDM), or their application to specific problem situations (Problem Situation). This is expressed in the following context statement:

Student C(I = 0, F = 0) SSDM

SSDM C(I = 0, F = 0) Problem Situation

Students are introduced to subject matter pertaining to the detail and process of various methodologies, *without* concern for specific usage scenarios. Following this activity, students will now possess the requisite knowledge of system/software development methodologies even if they may still not be aware of areas of application of specific methodologies. Thus, students will acquire explicit knowledge of methodologies, although they may still be unable to contextually apply them to targeted problem situations. This can be expressed as:

Student C(I = 0, F = 1) SSDM

5.3.3 Analyzing Context for LO-3

The students participate in guided learning exercises concerning the development of understanding and evaluation of problem situations while developing informed recommendations *without particular emphasis on solution strategies*. This phase of learning is governed by efforts to *credit local knowledge*.

Student C(I = 1, F = 0) Problem Situation

Problem Situation C(I = 0, F = 1) SSDM

5.3.4 Analyzing Context for LO-4

The students now undertake activities to iterate through *constraints* that bear upon a particular problem situation and formulate and develop *implementation processes* in accordance with all identified aspects of the problem domain. An early design decision by the course organizers required that the students employ the Aspect-Oriented Thinking (AOT) framework [Flint, 2006] for this purpose. Consequently, the choice of the AOT framework is a firm constraint (no alternatives are considered).

Student C(I = 1, F = 1) SSDM

SSDM C(I = 1, F = 1) Problem Situation

Problem Situation C(I = 1, F = 1) AOT

5.3.5 Analyzing Context for LO-5

The students are to undertake activities to effectively *communicate* and *collaborate* in order to actively arrive at an agreement on the identified intervention in a problem situation.

Student C(I = 0, F = 1) Communication and collaboration

5.4 SOLUTION SPECIFICATION FOR LO-5

The OAR process for formulating a solution specification for implementing communication and collaboration for LO-5 using the Moodle LMS is now presented [Chemboli, 2010a].

5.4.1 Initializing External Shelves

The external shelves with prototypes for designing the specification for realizing LO-5 are illustrated in [Figure 5.2](#).

In order to align the learning outcomes of COMP8120 with assessment criteria, the prototypes for assessment modes in Bloom's taxonomy are identified as relevant recipes for consideration. Recipes for the activities which are pertinent to LO-5 are given in the Outcomes external shelf. Written or verbal communication between students can either be synchronous (in real-time) or asynchronous.

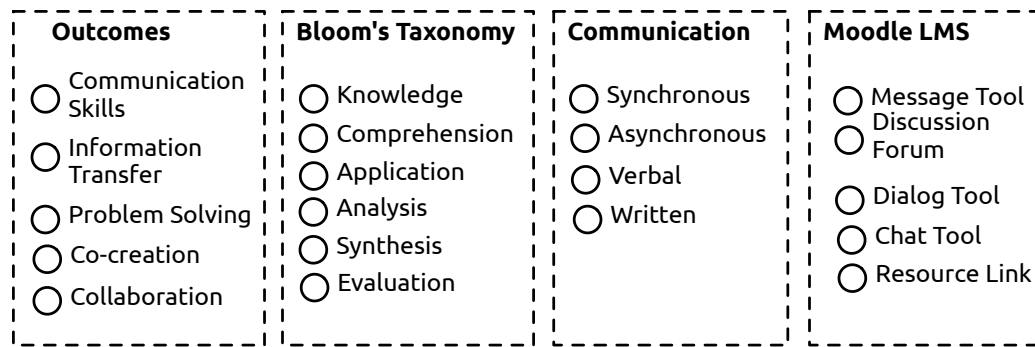


Figure 5.2: External shelves and prototypes.

These recipes are collected in the Communication external shelf. Since, LO-5 will be implemented using the Moodle LMS, an external shelf with resources and activities supporting communication and collaboration in Moodle is considered.

Although the original version of Bloom's taxonomy [Bloom, 1963] was used at the time of designing this course, there is nothing inherent in the OAR framework which mandates the kind of taxonomy used for aligning learning outcomes. The revised Bloom's taxonomy [Churches, 2008] can also be incorporated in a straightforward manner as another external shelf which provides additional prototypes for consideration in the solution specification.

5.4.2 Identifying Relevant Archetypes and Constraints

The next step is concern refinement for LO-5 in order to select the specific prototypes which can be imported into the problem domain shelf. LO-5 states the requirement for *active* communication and collaboration. The recipes for Communication and Collaboration are selected from the Outcomes external shelf. Two constraint recipes are also identified — *Synchronous and Written* from the Communication external shelf. The students are required to communicate and collaborate in a synchronous fashion using written tools (these constraints are imposed by the course designers). Utilizing Bloom's taxonomy, LO-5 can be aligned with assessment of Application skill. The students are required to utilize the communication platform to execute stakeholder engagement. Finally, prototypes for the activities in Moodle which support communication in a written form are selected. The resultant problem domain shelf is illustrated in [Figure 5.3](#).

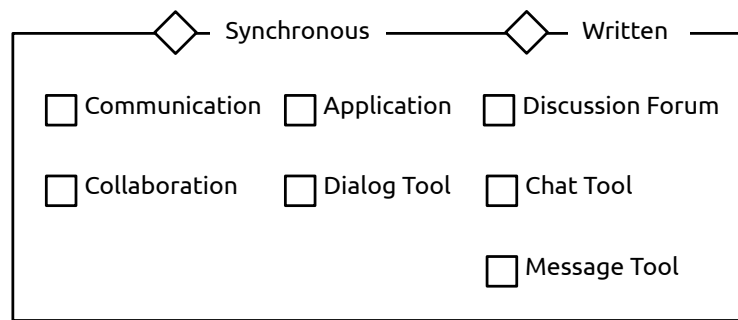


Figure 5.3: Archetype and constraint identification for LO-5.

5.4.3 Solution Shelf for LO-5

The process of Context Refinement for defining the solution specification for LO-5 (Figure 5.4) is outlined in the following specification statements:

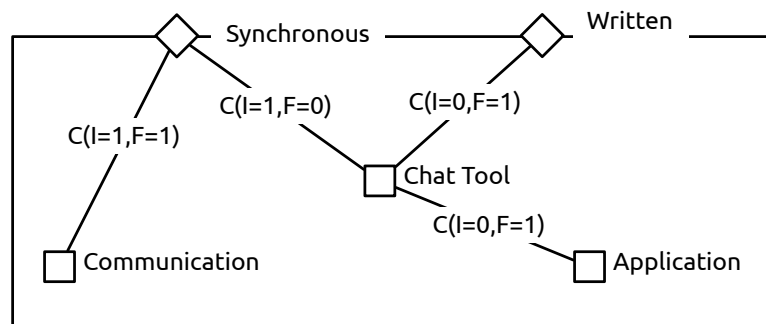


Figure 5.4: Solution specification for LO-5.

ss-1 The Communication archetype needs to be implemented in a *Synchronous* and *Written* fashion.

Synchronous $C(I = 1, F = 1)$ Communication

Communication $C(I = 1, F = 1)$ Written

ss-2 Although the Dialog Tool, Discussion Forum and Message Tool can be used for synchronous communication, they are not particularly suited for interactive text exchange between participants. Therefore, only the Chat Tool satisfies the requirement of synchronous communication, and is imported in the solution shelf.

Synchronous $C(I = 1, F = 0)$ Chat Tool

ss-3 The Chat Tool is selected because it is a writing tool.

Written C(I = 0, F = 1) Chat Tool

ss-4 The use of the Chat Tool will satisfy the requirement for assessing how the students apply the skills learnt via LO-5 because the course coordinators can assess the chat logs for significant contributions. This can be stated as:

Chat Tool C(I = 0, F = 1) Application

5.4.4 Implementing the Solution Specification

Finally, the solution specification for setting up a chat tool for satisfying LO-5 is implemented. This is achieved through either manual or automated translation using the information provided by the archetype for the chat tool. For instance, a Moodle workflow for setting up the chat activity in the LMS may be used [Chem-boli, 2010c]. This workflow can be specified in the chat tool recipe that is imported from the Moodle external shelf during the process of concern refinement. The use of the chat tool satisfies the course design requirement for LO-5 (synchronous and written communication). This is illustrated in [Figure 5.4](#).

5.5 SUMMARY AND CONCLUSION

The application of the OAR framework to contextualize course design by mapping the learning outcomes and intent of a course to its development and delivery is discussed in this chapter. Learning outcomes of a course are aligned with activities related to teaching and assessment by explicitly capturing the context of course design. The process is illustrated by an example specification to satisfy a learning outcome for synchronous and written collaboration between students in a software engineering course using the Moodle LMS.

This illustration brings out that OAR can be applied for workflow management across multiple disciplines — in the domain of education in this instance. The problem of communication and collaboration in realizing LO-5 is considered as a workflow. Capturing intellectual concerns of the problem domain as recipes in external shelves and formulating the problem-domain shelf and solution specification by concern and context refinement is clearly shown. The solution arrived at is unambiguous in terms of the concerns incorporated in the recipes.

6

MANAGING LARGE AND COMPLEX SYSTEMS

Neither your opinion of what users should think, nor my opinion of what users should think, matters as much as what users actually do think. Be a scientist, not a priest.

Shuttleworth [2012]

6.1	Large and Complex Systems	104
6.2	Some Characteristics of Large Systems	105
6.3	How complexity builds and escalates in large systems	106
6.4	Applying OAR to Complex Systems	107
6.5	The Ubuntu Platform as a Complex System-of-Systems	109
6.6	Capturing intellectual concerns for the Ubuntu ecosystem	109
6.6.1	Initializing External Shelves	110
6.7	Identifying Relevant Archetypes and Constraints	111
6.8	Solution Specification for Selecting the Default Music App	111
6.9	Utilizing a Solution Specification	113
6.10	Summary and Conclusion	117

The application presented in this chapter demonstrates the use of the OAR framework for the analysis, understanding and management of complex systems. The process of analysis, design and operation is considered as a workflow formulation, solution specification and implementation. An omniscient appraisal of all identified concerns in a system in terms of the component systems, the concerns within the component systems and the interactions between them is carried out; these concerns are subjected to the OAR processes of concern and context refinement for relating them to the modeling of the system and formulating a solution specification.

This chapter is organized as follows. Firstly, characteristics of large and complex systems and some issues of complexity are presented. Next, the manner in which OAR can be applied to complex systems is considered. This is followed by the application of OAR to an example in the Ubuntu software ecosystem — to analyze and work through the problem of selecting the default music app for Ubuntu

12.04 Long Term Support Support (LTS) distribution. Emergent behavior due to this selection is analyzed.

6.1 LARGE AND COMPLEX SYSTEMS

The study of large and complex systems, both engineered as well as naturally evolving, is gaining importance. This is mainly due to the wide-ranging impact of such systems on human society. Examples of such systems include large power networks, computer system networks, ecological systems etc. Though terms like large systems, complex systems, systems of systems and wicked problems are employed with varying and several connotations in the literature, complex systems are seen to possess a set of commonly recognized properties. Firstly, they consist of many component systems, interacting with each other and with the environment(s) within which they operate or exist. Secondly, they are open in nature in the sense that component systems may be added, modified, retired or replaced. Finally, the goals and objectives of such systems are very likely to change and evolve with time. Reviews of the salient features and concepts of complex systems in the context of systems of systems can be found in a number of references [Jamshidi, 2008; Sage and Cuppan, 2001; Sheard and Mostashari, 2009].

For the development of newer and more sustainable systems, it is usually necessary to analyze and understand the complex interactions in current systems. For this it is appropriate:

1. To be able to switch between detailed knowledge of the component systems and overall appraisal of the entire system. Current efforts to develop ontologies capturing 'complete' and 'universal' understanding of the entire system increase overall complexity, while simply providing minimally sufficient understanding across all interacting systems which may obscure details of component systems.
2. To develop the ability to treat each component subsystem and its constituents as a localized and single 'control and execution context' so as to enable system validation against desired goals and intent.

Inadequate support for the above features, coupled with uncertainties due to incompletely recognized or unrecognized interactions of the component systems, may further result in uncertainties in the outcomes of the processes orchestrated by the system, together with the increased likelihood of cost, effort and development time overruns in maintaining, enhancing, retiring and replacing systems.

6.2 SOME CHARACTERISTICS OF LARGE SYSTEMS

In so far as it could be determined, all the studies and attempts aimed at managing complexity are based on the implicit belief that proper design of a system can mitigate the issues/difficulties resulting from the complex interactions of the component systems comprising it.

According to Warfield [1994], a system is considered to be a large-scale and complex system only in relation to “human capacity to observe it, comprehend it, analyze it, steer it, amend it and tolerate it.” This concept of a complex system is based on consideration of both situational complexity and cognitive complexity, and any improvements to the management of complex systems should address both these kinds of complexity.

Systems that have technological underpinnings and have considerable sociological impact and interactions, are termed as socio-technical systems. Some of these, like forestscapes and water-body systems come into existence through a process of evolution. Some others, like human settlements, are designed and built (at least initially) and then continue to evolve [Alexander, 1964]. Socio-technical systems evolve through synergistic interactions between technology and people.

Technological systems that are designed and built can be separated into two main groups [Warfield, 1994]:

1. Systems that are designed entirely on the basis of well-established principles of science and engineering. *These systems can be validated against the standards of knowledge of science. Their failures are generally due to components and process parameter errors/inaccuracies and can be well-characterized. The impact of such failures can be managed well in most cases.* Examples of such systems are radio and television broadcast systems, chemical plants etc.
2. Systems that are designed and built on the basis of perceptions and models of the needs of the user and properties of the system, like computer software, programming languages etc. *Some of these systems may additionally include a number of entities of the first kind. For such systems, reference against primary standards of science is less direct because their reference is through the models used in the design.* Examples of such composite systems are information systems, management support systems, expert computer systems, hospital and health care systems, nuclear plants and banking organizations etc. *The behavior of such systems is similar to socio-technical systems because their satisfactory performance depends on the synergistic interaction of their component systems.*

Basic scientific exploration and research of complex systems consists of recognizing a system as complex, recognizing possible manifestations of such complexity

and formulating the concepts and models that may be used in their description. Theories of catastrophe [Arnold, 1986], chaos and entropy [Mitchell, 2009] provide directions to the efforts at gaining knowledge in this domain.

6.3 HOW COMPLEXITY BUILDS AND ESCALATES IN LARGE SYSTEMS

A typical scenario of managing complexity consists of careful design of the processes and working environment and assigning specialized roles to properly trained personnel. The situational complexity of a system arises due to interactions amongst the component systems as well as interactions with the environment. Added to this will be features of cognitive complexity — as felt and perceived by the personnel managing the system. As depicted in Figure 6.1, additional resources inducted into the system for the purpose of managing complexity, themselves, will escalate complexity by creating new linkages [Warfield, 1994]. Thus, managing a complex system should focus on control of situational complexity, at the same time keeping the scope of management within the cognitive abilities of the people managing the system.

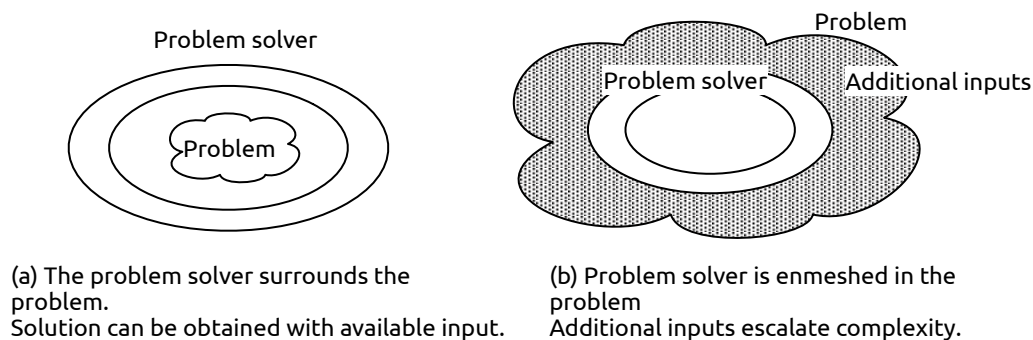


Figure 6.1: Escalation of problem complexity when additional inputs are provided.

Effective management requires analyzing and understanding the design of the system in terms of the concepts, models and implementations involved. Recognizing the concerns that are vaguely understood but relevant to the outcomes of the system will enable the planning of strategies for managing failures due to these concerns. The OAR framework has features that are well-suited to address these problems in large systems, as will be described in the next section. The framework supports the maxim, *what is understood with referential validity is science, and what is modeled and designed in terms of the science is technology.*

6.4 APPLYING OAR TO COMPLEX SYSTEMS

In the OAR framework, the process of analysis, design and operation of a system is considered as a workflow formulation, solution specification and implementation. In relation to complex systems, a particular outcome or application of the system is considered to be the problem scenario and the OAR process is applied to it.

Complex systems integrate many interacting (diverse) disciplines. As a result of this diversity, forcing a common ontology for entire systems of systems may result in loss of valuable domain insights. Conversely, a highly focused exploration of individual system domains may also lead to a loss of generality in the nature of systems understanding. Thus, a common ground between too much detail of individual domains and too limited a view of the entire system, is highly desirable. Current efforts in this direction [Staab and Studer, 2009] are focused on developing and building comprehensive language and concept interchange formats for “universal use” — ontologies for universal acceptance. Large ontologies have the side-effect of further adding to the net system complexity owing to the difficulty of arriving at a common understanding that may be universally applicable across many/all the disparate component systems. Most strategies employed [Jordan and Cicortas, 2008; Sahoo, Sheth, and Henson, 2008; Truong et al., 2005] aim to create ontologies which furnish minimally sufficient understanding across the entire system, thus losing significant detail of intellectual concerns in the component systems.

A way forward is to make use of localized ontologies to capture and represent the individual ‘silos’ of knowledge in domains which constitute the large system and develop the interchange processes for only that subset of the intellectual concerns which are relevant for the application of the complex system. These localized ontologies need not be complete in the sense of universal applicability. But they will unambiguously capture all the system artifacts along with information about the context of their application.

As a result, large systems of systems may now be viewed as an aggregation of numerous localized ontologies, each of which can be further adapted, evolved, retired and replaced as appropriate to changes in the circumstances in which the systems operate and interact.

Localized ontologies are realized by shelf management in the OAR framework. Intellectual concerns across component subsystems are captured in recipes by the process of concern refinement. The collection of external shelves represent all available interdisciplinary knowledge which can be used to manage the complex system and its component systems. Interactions between component systems may

themselves be the subject matter of further external shelves and the problem domain shelf.

Large and complex systems are typically characterized by the absence of a single control and execution context. This often results in difficulty in managing the component systems and the interactions between them. Using the OAR framework, context relations within and between individual component systems can be characterized by:

1. Strong influence and high firmness ($C(I = 1, F = 1)$), i.e., the context relations between the recipes within each system and between systems are significant and well-understood. This means that the interaction between the systems will not result in any unexpected behavior.
2. Strong influence and low firmness ($C(I = 1, F = 0)$), i.e., the concerns affecting the behavior of the component systems are recognized as significant, however the recipes are not well-formed. As a result, unintended system behavior cannot be ruled out.
3. Weak influence and low firmness ($C(I = 0, F = 0)$), i.e., the system interactions are neither properly recognized nor understood — they are only suspected to be present, and may produce unpredictable and unexpected effects.
4. Weak influence and high firmness ($C(I = 0, F = 1)$), i.e., although the recipes are well-formed, their interaction is either not known or is considered insignificant.

The last two context relations indicate the need for further research in identifying the particular influences.

Context refinement thus enables the system to be considered as an aggregate of component systems each with its goals and intent, and then working out a linkage between them. This facilitates local control and execution contexts for both the component systems and the interactions between them. The *solution specification*, an artifact arrived at by utilizing localized ontologies via shelf management, and concern and context refinement processes, corresponds to the outcome of the application of the system.

The nature of omnispersive analysis:

1. Enables better management of system development lifecycle by taking into account component systems and their interactions;
2. Facilitates verification and validation of the underlying intellectual concerns in the component systems, their interactions and thus of the overall system;

3. Reduces the cognitive burden on the human interacting with the system by organizing the intellectual concerns as recipes in accordance with concept level, model level and implementation level.

6.5 THE UBUNTU PLATFORM AS A COMPLEX SYSTEM-OF-SYSTEMS

According to the criteria outlined by Abbott [2006] the Ubuntu platform ecosystem qualifies as a complex system. It is open and consists of numerous interacting and continually evolving component systems; which are added, modified, replaced or retired over time.

This illustration of the application of the OAR framework centers on analyzing and working through the problem of selecting the default music app for the Ubuntu 12.04 Long Term Support (LTS) distribution. The problem involves the interplay of a number of engineered, societal, legal and various other systems, each with its own cadence and pace of change and development and the influence of changes in component systems is felt locally as well as in other systems within the wider platform ecosystem.

The selection of the default music app for an operating system distribution may not result in 'disaster' in the event of failure. Nevertheless, it amply illustrates the manner in which all the component systems and their interactions need to be examined in order to arrive at a solution specification using the OAR framework. Thus this illustration provides insight to the management of localized ontologies developed through the OAR process, and also demonstrates how the difficulty of not having a single control and execution context can be mitigated because the OAR solution specification provides for validation against desired intent and goals.

6.6 CAPTURING INTELLECTUAL CONCERNS FOR THE UBUNTU ECOSYSTEM

Identified knowledge in different domains within the Ubuntu ecosystem is captured in different recipes which are managed through external shelves in the OAR framework. Depending upon the granularity of the problem objective, the choice of the external shelves varies.

Recipe details incorporate outcomes of Internet Relay Chat (IRC) logs, document blueprints, Ubuntu Developer Summit proceedings, discussion forums and project guidelines.

6.6.1 Initializing External Shelves

The external shelves considered in deciding the default music app are shown in [Figure 6.2](#).

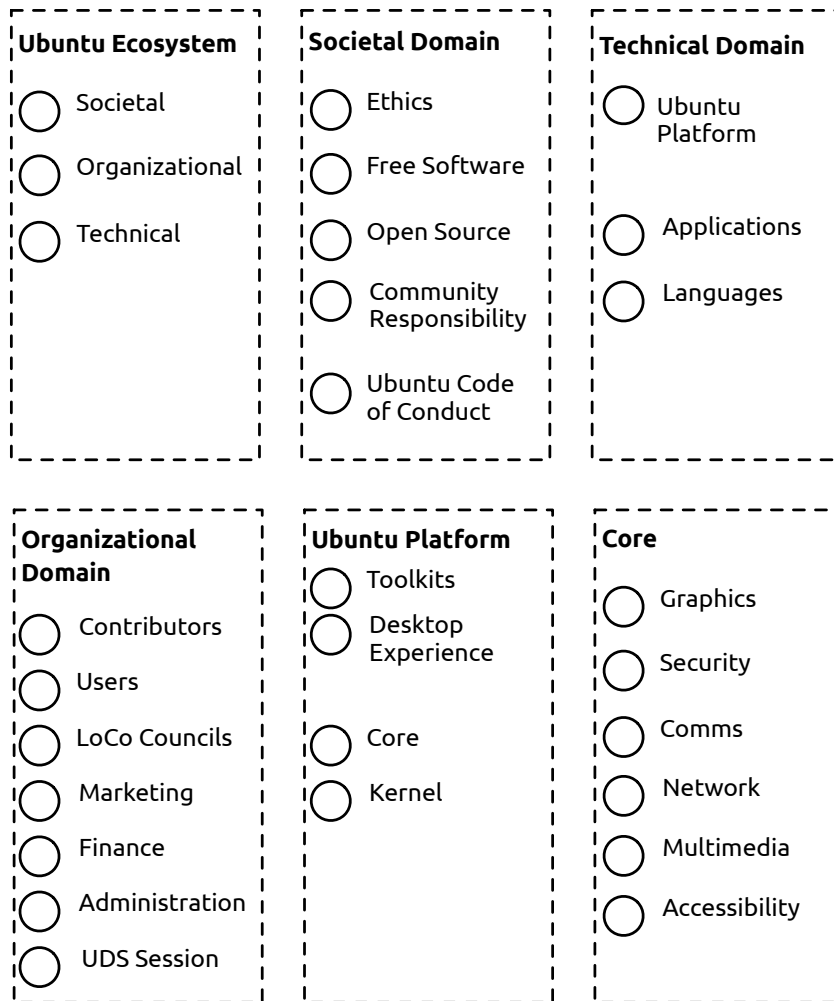


Figure 6.2: External shelves and recipes for deciding the default music app.

The idea of localized ontologies can be seen in action here in selecting shelves that are adequate for steering a solution specification.

6.7 IDENTIFYING RELEVANT ARCHETYPES AND CONSTRAINTS

The process of concern refinement is now carried out in order to select the specific prototypes which can be imported into the problem domain shelf. As a result, the prototypes for LTS, ISO Size, Licensing and Release Date are recognized as constraints. The problem domain shelf is populated with further prototypes which may now be considered as archetypes owing to their relevance to the problem domain. This is shown in [Figure 6.3](#).

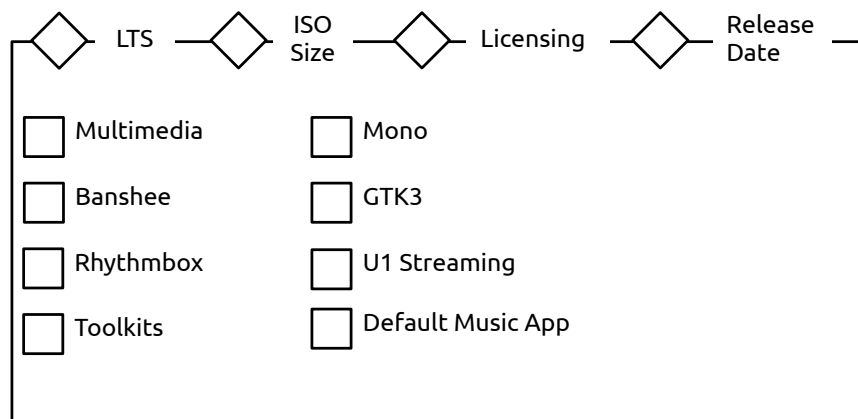


Figure 6.3: Archetype and constraint identification in the problem domain shelf.

6.8 SOLUTION SPECIFICATION FOR SELECTING THE DEFAULT MUSIC APP

A solution specification is arrived at by the OAR process of context refinement. Accordingly, context refinement is performed to define a solution specification for selecting the default music app in Ubuntu 12.04 LTS. This process is carried out as follows:

ss-1 A Long-Term Support (LTS) release of the Ubuntu Operating System is of five years duration. In addition, each LTS is guaranteed to provide a dependable upgrade path to the next LTS release. Hence, it can be seen that Ubuntu 12.04 is subject to the constraint of rigid conformance to LTS guidelines, and is required to support the platform technologies for the entire duration.

LTS C(I = 1, F = 1) 5 Years

Ubuntu 12.04 C(I = 1, F = 1) LTS

ss-2 The Ubuntu platform distribution is built as a live and installable CD which has a constraint of maximum size (< 700 mb).

ISO Size C(I = 1, F = 1) <700mb

ss-3 The release date is decided as per policy and is for most intents and purposes, non-negotiable. Therefore, this is a strict constraint.

Release Date C(I = 1, F = 1) 2012-04-26

ss-4 The MonoGtk3 port is still under development and will not be ready before the Release Date. The Banshee music app is developed on the MonoGtk stack. Porting the app to Gtk3 cannot progress until the MonoGtk3 toolset is stable. On the other hand, the development of Rhythmbox is not dependent on the status of the MonoGtk3 port, since it is coded in the C programming language with libgobject bindings. Accordingly, the context states are specified as:

MonoGtk3 C(I = 0, F = 0) Release Date

MonoGtk3 C(I = 1, F = 1) Banshee

MonoGtk3 C(I = 0, F = 0) Rhythmbox

ss-5 One of the goals of the LTS distribution is substantial migration of the platform to the GTK+3 toolkit. This decision introduces a preference to include a GTK+3 app where possible if it provides reasonable feature parity with other alternatives. The Rhythmbox media player has been completely ported to GTK+3. As seen in SS-4, MonoGtk3 still requires further development.

GTK3 C(I = 1, F = 1) Ubuntu 12.04

GTK3 C(I = 1, F = 1) Rhythmbox

MonoGtk3 C(I = 0, F = 1) GTK3

ss-6 The Ubuntu One cloud service is fully implemented as a plugin for Banshee. A Rhythmbox plugin is currently in development, and cannot yet be considered firm.

U1 Service C(I = 1, F = 1) Banshee

U1 Service C(I = 1, F = 0) Rhythmbox

ss-7 In terms of licensing, both media players satisfy the criteria for Free and Open Source software.

Licensing $C(I = 1, F = 1)$ Banshee

Licensing $C(I = 1, F = 1)$ Rhythmbox

With these constraints, the solution specification which is illustrated in [Figure 6.4](#) is obtained.

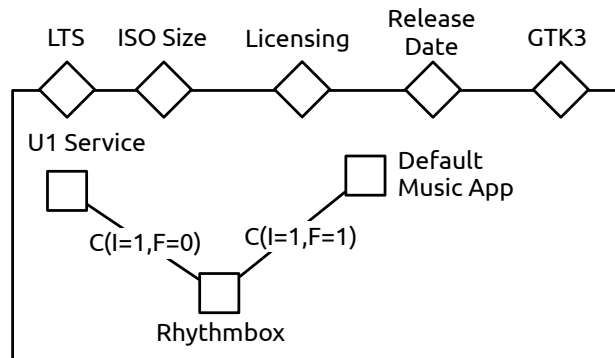


Figure 6.4: Solution shelf for the default music app specification.

6.9 UTILIZING A SOLUTION SPECIFICATION

The solution shelf illustrated in [Figure 6.4](#) is one instance of the analysis and intervention for the selection of the default music app in the Ubuntu 12.04 LTS system. Subsequently, this recipe (solution specification) may be treated as a member of another external shelf of recipes within the entire Ubuntu ecosystem. There is no requirement for recipe membership to be unique to a particular shelf, and depending on the granularity of analysis of the component system, the same recipe may be utilized across multiple shelves.

Thus, along with the problem domain shelf, the solution shelf also encapsulates a localized ontology which captures the intellectual concerns influencing the selection of the default music app within the Ubuntu 12.04 LTS system.

Moreover, multiple paths of context refinement may provide a family of related solution specification recipes populating related external shelves. For instance, a DVD respin of the Ubuntu 12.04 LTS system may include the Banshee media app since it is free of the ISO Size limitation. This will have further implications for other component systems and their interactions and behavior. Hence, even though shelf categorization need not be unique, a solution specification, or for that matter, any recipe is unambiguous in translation even though multiple pathways of implementation and further processing may exist depending on the system context. The solution specification of [Figure 6.4](#) provides a clear pathway for analyzing the interactions of other component systems which may

be affected as a result of the constraint to remove the MonoGTK3 binding from the standard distribution. This approach provides the ability to treat each component subsystem and its constituents as a localized and single ‘control and execution context’ while making it possible to conveniently switch between detailed component system analysis and an overall appraisal of the entire system.

Unintended side-effects may evolve due to the interactions between the recipe for the default music app and the rest of the Ubuntu ecosystem. This is illustrated by considering how the solution specification in [Figure 6.4](#) affects the selection of the default app for note-taking.

The external shelves (localized ontologies) for App Categories and some Default Apps are depicted in [Figure 6.5](#). The solution specification for the default music app is now a recipe in the default apps shelf.

In order to analyze how the choice of Rhythmbox as the default music app affects the selection of the default note app, concern refinement is performed to populate the problem domain shelf illustrated in [Figure 6.6](#).

Context refinement yields the following solution specification statements.

ss-8 It is a requirement for the note taking app to synchronize with the U1 Sync service. Tomboy supports U1 Sync, while GNote lacks this functionality.

U1 Sync C(I = 1, F = 1) Tomboy
U1 Sync C(I = 1, F = 0) GNote

ss-9 As seen in SS-4, Rhythmbox does not have a dependency on the Mono framework, which is not yet fully ported to Gtk3. In addition, GNote is also independent of the Mono stack, while Tomboy has a contextual dependency on the same.

Mono C(I = 1, F = 1) Tomboy
Mono C(I = 0, F = 0) GNote

ss-10 Consequently, the emergence of a contextual relationship between the choice of the Default Music App and the Default Note App can be seen.

Default Music App C(I = 1, F = 0) Default Note App

ss-11 Following from SS-8, since GNote does not incorporate critical synchronization features, it is not suitable for selection in the LTS distribution.

GNote C(I = 1, F = 0) LTS

ss-12 From SS-10, SS-4 and SS-2, it can be seen that the inclusion of Tomboy requires the entire Mono stack, affecting the ISO size.

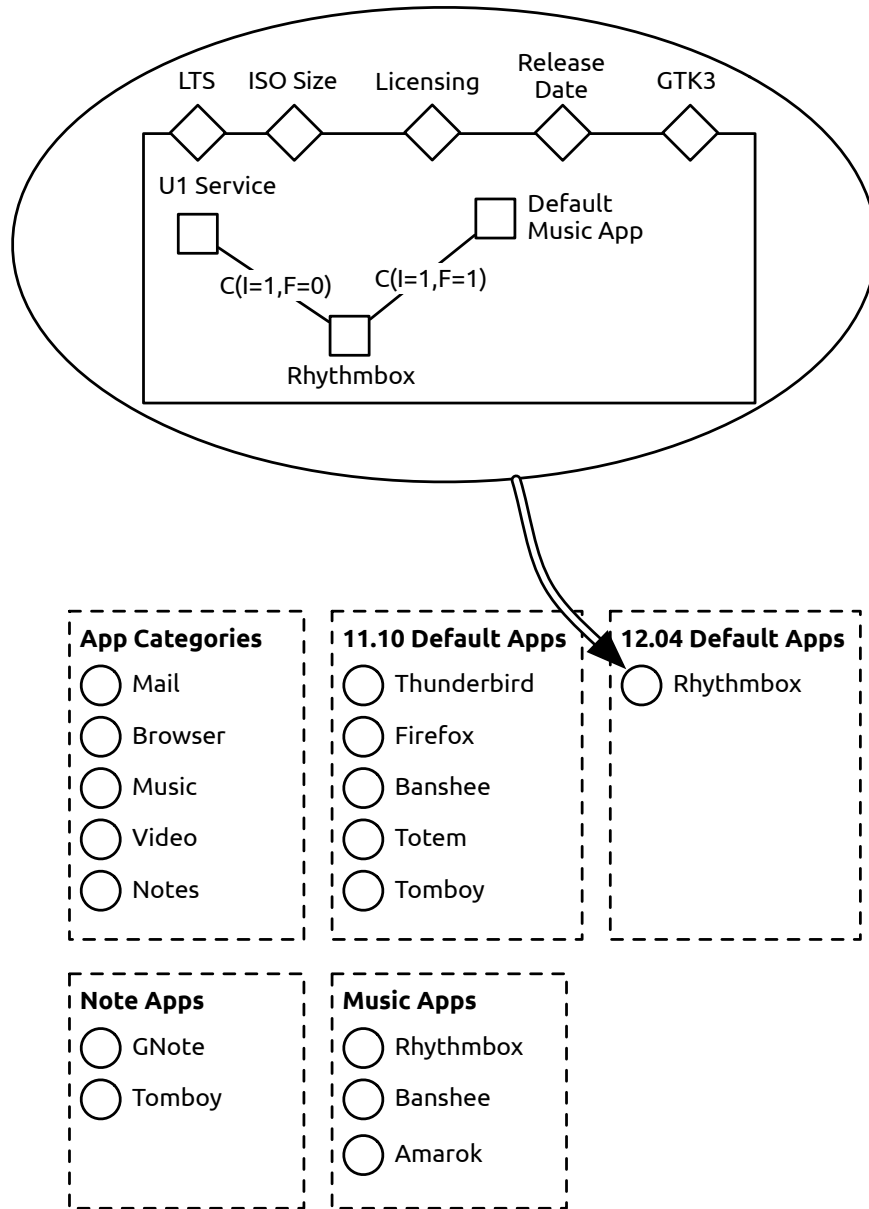


Figure 6.5: Local ontology for default apps.

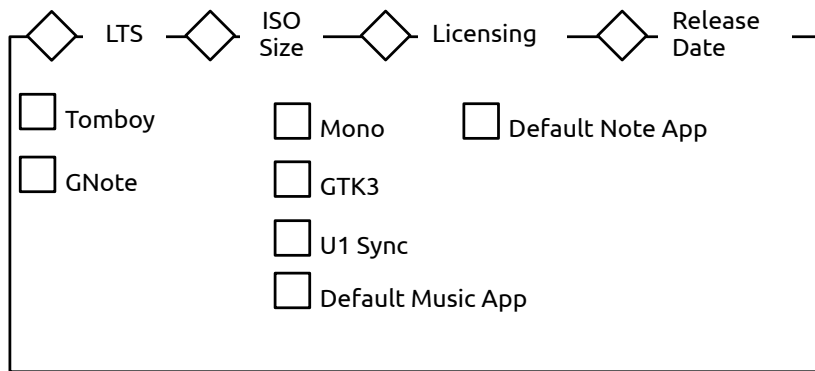


Figure 6.6: The problem-domain shelf for selecting the default note app.

ISO Size $C(I = 1, F = 1)$ Default Note App

ISO Size $C(I = 0, F = 0)$ Tomboy

ss-13 Thus, the choice of the Default Music App results in the exclusion of Tomboy. The relation between these two recipes is a strong constraint.

Default Music App $C(I = 1, F = 1)$ Tomboy

ss-14 In light of SS-8, SS-9, SS-12 and SS-13, neither Tomboy nor GNote can be selected as the default note app for the LTS distribution.

Tomboy $C(I = 0, F = 0)$ Default Note App

GNote $C(I = 0, F = 0)$ Default Note App

Default Note App $C(I = 0, F = 0)$ LTS

In this case, owing to the choice of the Default Music App, the Ubuntu 12.04 LTS distribution will not ship with a default note app. The note app can be seen to be a vaguely defined and unimplemented prototype.

The corresponding solution specification is illustrated in Figure 6.7.

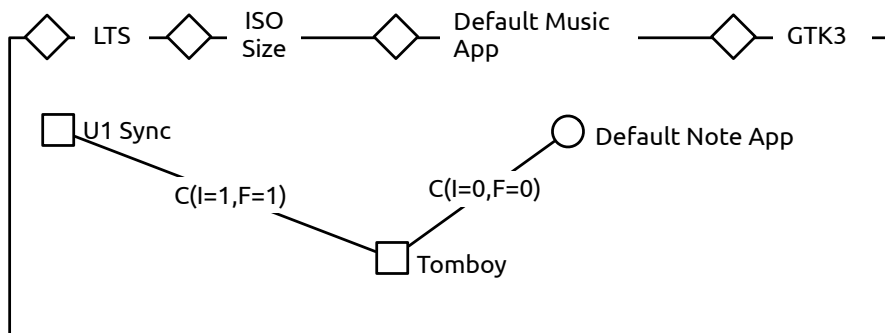


Figure 6.7: In this recipe, the Ubuntu 12.04 LTS distribution does not ship with a default note app.

ss-15 On the other hand, if the presence of a default note app is considered as a strong constraint, then in spite of the arguments against it, the only valid choice is GNote. The resulting solution specification cannot be considered an archetype owing to the ambiguity in the choice of GNote (the app lacks critical synchronization features).

Default Note App $C(I = 1, F = 0)$ GNote

As a result, GNote cannot be considered an archetype for the resultant solution specification (Figure 6.8).

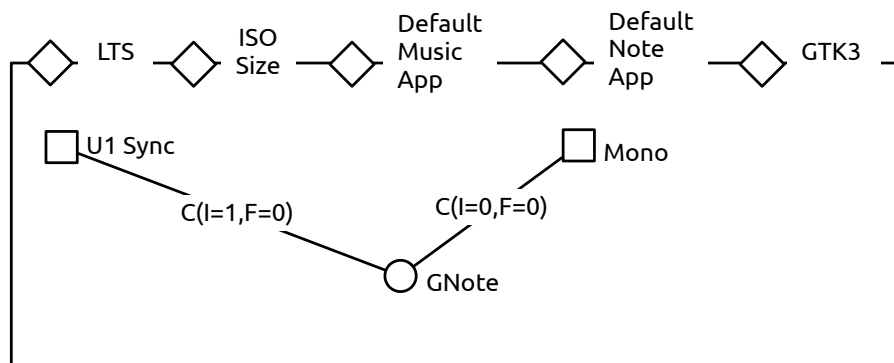


Figure 6.8: In this recipe, the Ubuntu 12.04 LTS distribution can ship with GNote as the default note app even though it lacks critical synchronization features.

6.10 SUMMARY AND CONCLUSION

The application of the OAR framework for the analysis and management of large and complex systems has been presented in this chapter. The nature of large systems of systems has been briefly considered, and it has been particularly identified that the analysis, understanding and management of complex systems can be further improved by utilizing the OAR framework to develop:

1. Localized ontologies to facilitate switching between detailed knowledge of the component systems and an overall appraisal of the entire system.
2. The ability to treat each component system and its constituents as a local and single control and execution context, thus enabling validation of the component systems against overall goals and intent.

The process of selecting the default music app for the Ubuntu 12.04 LTS distribution has been considered to illustrate the application of OAR to systems

of systems, and a solution specification for this is obtained using the processes of concern and context refinement. Localized ontologies and fixing the control context enable the exposure of side-effects that might evolve across the Ubuntu platform due to a particular solution specification for the default music app.

In the OAR framework, the process of analysis, design and operation of a system is considered as a workflow formulation, solution specification and implementation. An omniscient appraisal of large systems of systems enables the capture of all identified intellectual concerns in component systems as individual recipes. A problem scenario can then be reformulated in terms of relevant recipes encapsulating the concerns in the component systems, their mutual interactions and the influence they have on one another. The management of these recipes (prototypes, archetypes and constraints in the problem scenario) in shelves provides the ability to organize and use localized ontologies for component systems, which are sufficient for analyzing, formulating and obtaining solution specifications for the given application. Those concerns that are vaguely understood but may be relevant to the outcomes of the system can be recognized.

The OAR processes of concern and context refinement facilitate the verification and validation of intellectual concerns in component systems as well as verifying the interactions between them and formulating their behavior and outcomes as solution specifications in terms of, and conforming to, the OAR recipes. This provides the ability to treat component subsystems as localized control contexts within the larger system scope.

Further studies are needed to fully explore the scope and extent of the benefits obtained by applying the OAR framework to the analysis and design of complex systems.

Part IV

Conclusion

7

SOMA: OAR TOOL PROTOTYPE

[It] is almost always incorrect to begin the decomposition of a system into modules on the basis of a flowchart.

Parnas [1972]

7.1	Soma: A Tool for Simple Omnispective Analysis and Reasoning	122
7.1.1	Initialization	122
7.1.2	Building the Problem–domain Shelf	123
7.1.3	Contextualization	124
7.1.4	Practical Considerations in Soma	124
7.2	Product Vision and Goal	126
7.3	Architecture Vision and Sprint Planning	126
7.3.1	Architecture Vision	127
7.4	Soma Development	128
7.4.1	Soma Sprint 1	128
7.4.2	Soma Sprint 2	135
7.5	Summary and Conclusion	137

The necessity of tooling and automation support becomes increasingly apparent as the scale and magnitude of problems being tackled increases. For instance, the use of tractors and combines becomes necessary in order to facilitate large-scale cultivation within manageable timeframes. While manual cultivation, without either the benefit or speed of tools, is certainly doable (not unlike in the pre-mechanization era), the task of tilling thousands of acres before the sowing season commences may simply not be feasible without automated processes. Here, the need for automation and tooling support does not arise from any inherent limitations in the concepts and processes of cultivation — concepts and ideas always scale. Any limitations of scale emerge only through the limitations of human capacity and the tools employed. Developing suitable automation and tooling infrastructure bolsters the capacity to carry out large tasks within reasonable timeframes.

Analogously, while the application of OAR to workflows incorporating significantly large number of concerns can be carried out manually, it is expedient to

avail of tool support. To this end, the design and implementation of Soma, an OAR tool prototype [Chemboli, 2012b] is presented in this chapter. This development has been carried out using the Scrum approach [Pham and Pham, 2012; Schwaber and Sutherland, 2011]. The origami iris flower workflow (Chapter 4) is used to illustrate the formulation of a solution specification using Soma. Some directions for further engineering and development are also indicated.

7.1 SOMA: A TOOL FOR SIMPLE OMNISPECTIVE ANALYSIS AND REASONING

The workflow for a research activity starts with the examination of the problem situation and the scientific domain of the problem by omnisppection and analysis and abstracting the concerns of the research domain as suitable recipes. In light of the discussion of the OAR framework in Part II and the illustrations presented in Part III, the following requirements can be identified for a tool for supporting OAR.

The ukes and uke groups that are necessary for formulating the research problem are identified and formulated in terms of recipes which are abstracted in a suitable form at the concept, model or implementation levels. Each recipe is given a Universally Unique Identifier (UUID) for unambiguous reference and use.

The next requirement is the management and organization of recipes in shelves. All available recipes are categorized and assigned to various external shelves.

The tool should support the definition and formulation of a problem domain shelf by concern refinement. Here, the recipes which are considered relevant to formulating the research problem are added to the problem–domain shelf. Archetypes and constraints are identified by examining these prototypes in light of the scientific basis of the research problem.

Finally, the tool should support the specification of a solution shelf by contextualizing the archetypes and prototypes in the problem–domain shelf, subject to the limitations imposed by the constraints.

In addition to the above requirements, provision for editing existing recipes and shelves is a requirement for carrying out robust shelf management.

7.1.1 Initialization

A convenient way of visualizing the development of the workflow is to depict the progression of the workflow on a canvas. It may be noted that the concept

of recipe is generic and various shelves employed may themselves be considered as recipes which are identified with their UUIDs. As described in [Section 3.15](#), in case there are no preexisting external shelves with relevant recipes, the first step consists of bootstrapping external shelves. Subsequently, the initial state of Soma ([Figure 7.1](#)) can consist of external shelves with a number of recipes which are suitably characterized and identified.

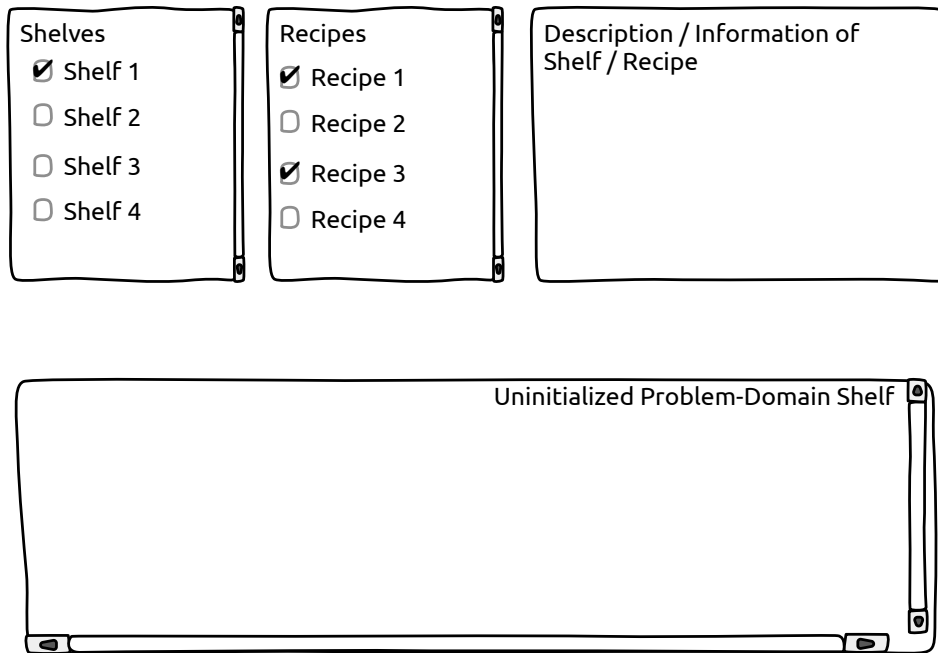


Figure 7.1: The initial state of Soma with several external shelves and an empty problem-domain shelf canvas.

At this stage, the problem-domain shelf is an empty canvas.

7.1.2 Building the Problem-domain Shelf

Based on the nature of the research, a suitable name is provided for the problem-domain shelf and a new recipe is created with this name and assigned a UUID. Next, the problem-domain shelf is populated with recipes selected from the external shelves by the OAR process of concern refinement as depicted in [Figure 7.2](#).

All recipes are initially considered as prototypes; on evaluation by reasoning some of them are identified as archetypes in the problem-domain.

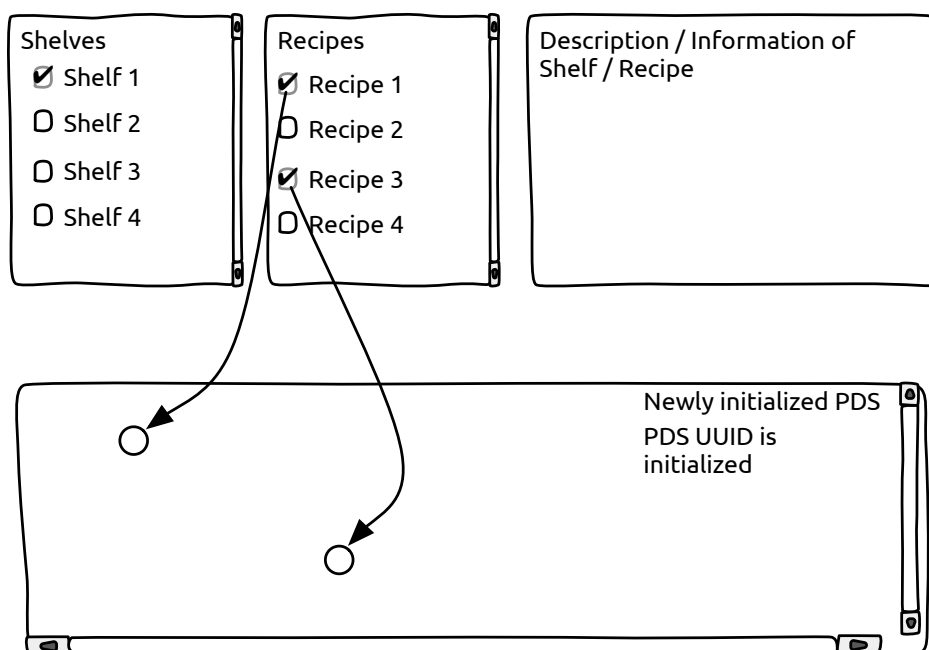


Figure 7.2: The newly initialized PDS is populated

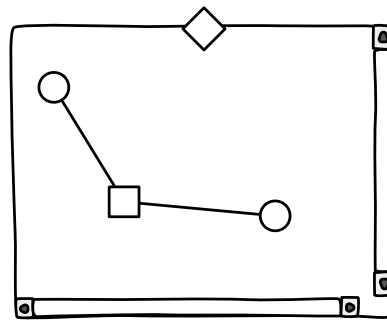
7.1.3 Contextualization

Conversion of the problem–domain shelf into a corresponding solution shelf begins with identifying the sequence of using the recipes and articulating the contextual relations between them. Recipes that act as constraints in arriving at the solution specification are identified by the OAR process of context refinement. Although constraints may pertain to specific recipes or groups of recipes, they are required for the entire solution specification to emerge. Placing the constraints on the solution shelf boundary (as used in the illustrations in [Part III](#)) provides a visual simplification. In practice, such a simplification may imply several different paths of contextual relations as illustrated in [Figure 7.3](#).

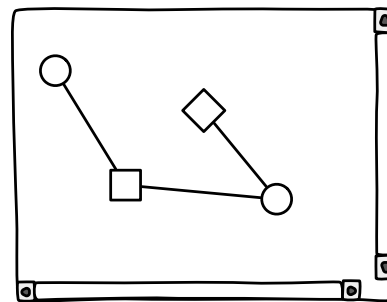
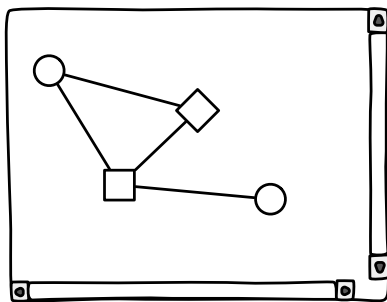
Hence explicit elaboration of constraint context may be sometimes required to avoid any possible ambiguity in the solution specification. No ambiguity arises if the constraint imposes universally on all the recipes in the solution specification.

7.1.4 Practical Considerations in Soma

In Soma, the processes of concern and context refinement, characterizing recipes as archetypes and constraints are carried out interactively by the workflow practitioner.



(a) Solution Shelf representation with Constraints on shelf boundary



(b) The visually simplified representation in (a) may further correspond to various elaborations, two of which are illustrated here.

Figure 7.3: Possible ambiguity due to visual simplification of constraint representation in a solution shelf.

All concerns — recipes, shelves, contextual relations — are represented by recipes in Soma. This may result in rapid increase in the number of defined recipes. This *recipe proliferation* may be managed by incorporating appropriate categorization, tagging, search and retrieval processes. Since OAR is an epistemic and conceptual framework, and scales infinitely, any limitations of scale in tooling will arise solely from restrictions imposed by the particular choice and nature of the underlying database frameworks employed.

A practical approximation in the OAR framework is the consideration of context in OAR in terms of four discrete pairs of Firmness (F) and Influence (I) (Section 3.16). The same practical approach is adopted in Soma to represent contextual relationships.

7.2 PRODUCT VISION AND GOAL

The product vision of Soma is to create a simple and representative application for handling the processes of recipe creation, shelf management, and concern and context refinement. The aim of the tool is to provide a reference implementation of tool principles for OAR. The overall goal of Soma is to illustrate the processes of OAR as simply as possible.

Accordingly, the scope of Soma can be stated as:

WHO (ARE THE USERS) Tool developers and users of simple OAR workflows.

WHY (PURPOSE) Illustrate a reference approach to automating and tooling the OAR framework.

WHAT (ESSENTIAL FEATURES) Recipe and Shelf management for the OAR framework.

7.3 ARCHITECTURE VISION AND SPRINT PLANNING

Using visual requirements gathering, the requirements for the Soma tool prototype which have been stated earlier in this chapter can be represented in the tree and forest analogy illustrated in Figure 7.4.

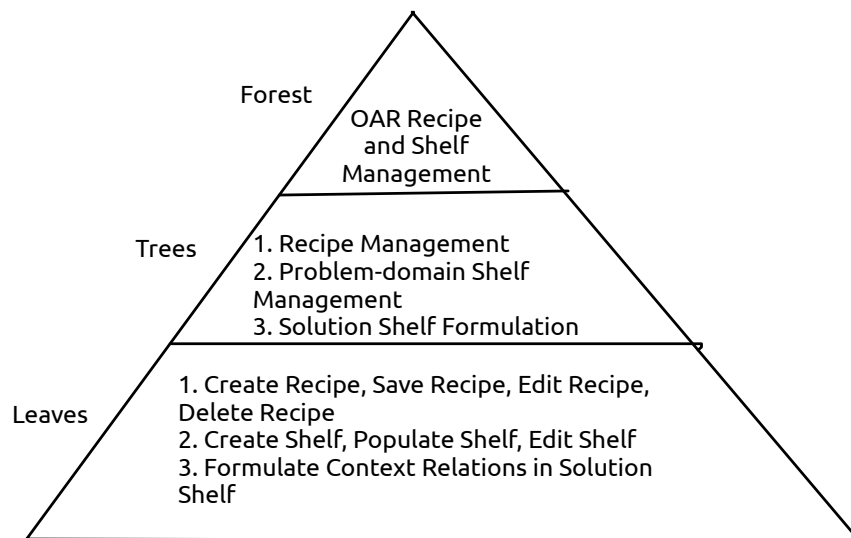


Figure 7.4: The requirements pyramid for the Soma OAR tool prototype.

7.3.1 Architecture Vision

Using the Feature Cluster process [Pham and Pham, 2012], the architecture cluster for Soma is formulated as shown in Figure 7.5.

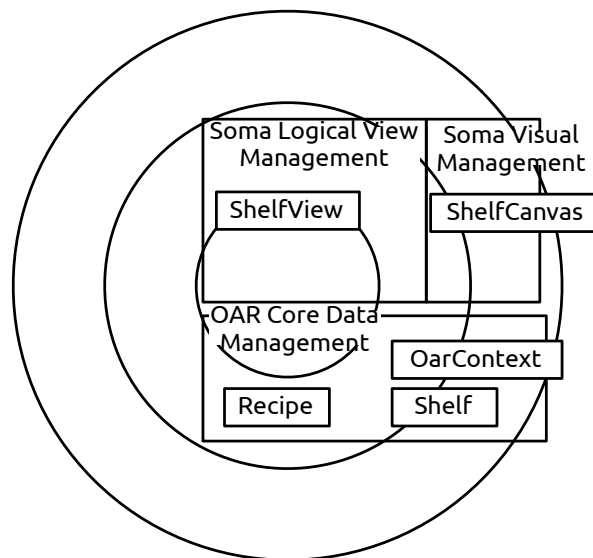


Figure 7.5: Sliced view of the Soma architecture cluster.

This corresponds to the data architecture model of Soma illustrated in Figure 7.6.

As shown in Figure 7.6, a three-layered data architecture model is utilized to semantically separate the representation of data in OAR and its visualization.

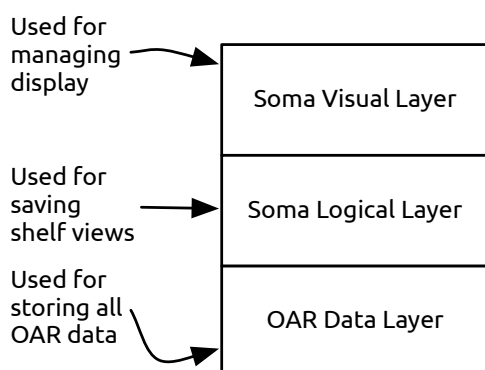


Figure 7.6: Data architecture model of Soma. Storage of OAR data is independent of its representation in the Soma Logical Layer. The Soma Visual Layer can be customized to generate different outputs.

The development of the Soma prototype is carried out using a horizontal sliced approach across two sprints. In the first sprint, development of features concerning the management of OAR Core Data and Logical View Management in Soma is carried out. These features are further developed together with Soma Visual Management features in the second sprint.

By following this approach, rudimentary Logical View Management features can be implemented before OAR Core Data Layer is fully realized. As shown in [Table 7.1](#), this further enables the commencement of feature development for Visual Management in the second sprint with incremental code drops.

7.4 SOMA DEVELOPMENT

The Soma implementation prototype is developed for the Ubuntu 12.04 LTS platform [Canonical, 2012]. This is implemented in Python 2.7.x [Python Software Foundation, 2012] using the PyGObject bindings for the GTK+3 toolkit [The GNOME Project, 2012]. The data store for the application is managed by a local instance of an SQLite database [SQLite, 2012].

7.4.1 Soma Sprint 1

The development of the data model for Soma and a corresponding workflow model are undertaken in Sprint 1.

The organization of data models for Sprint 1 is shown in [Figure 7.7](#).

The OAR data layer enables the storage and manipulation of recipe and shelf definitions in a manner independent of their visual representation. The SQLite

Forest	Trees	Leaves
OAR Recipe and Shelf Management		
	Release 0.1	
	Sprint 1	
	Recipe Management Shelf Management	Create, Save Recipe Create Shelf, Populate Shelf
	Sprint 2	
	Shelf Management Solution Shelf Formulation	Save Shelf, Edit Shelf Formulate Context Relations

Table 7.1: Soma release and sprint planning.

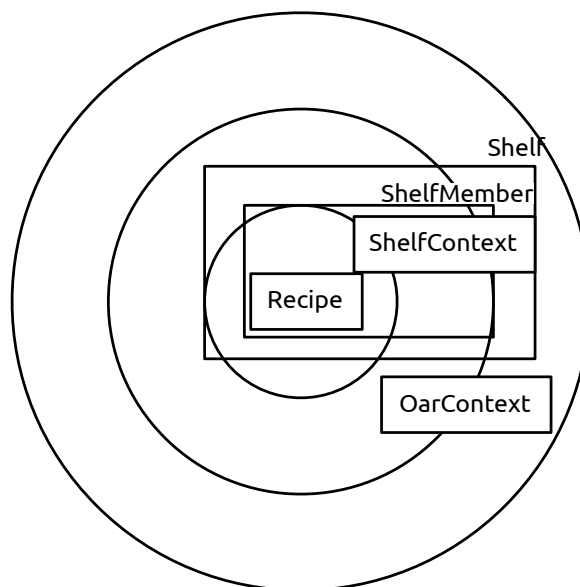


Figure 7.7: Soma data ring during Sprint 1.

database stores recipe and shelf information, along with the details of the Soma Logical Layer and the Soma Visual Layer. Each shelf in Soma corresponds to a recipe which stores the shelf definition, and is thus a recipe itself.

The data model for Soma is illustrated in [Figure 7.8](#).

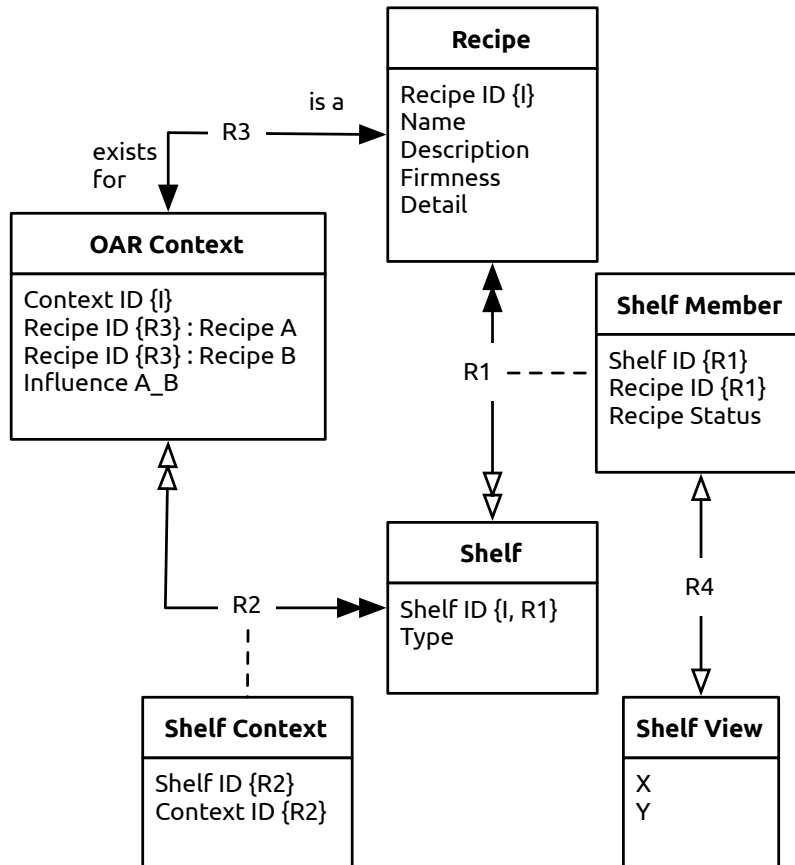


Figure 7.8: Soma data model.

Creating a Recipe

The layered data architecture for Soma enables the management of OAR data independently of its visual representation.

The opening screen of the tool prototype is illustrated in [Figure 7.9](#).

The user-interface of Soma is divided into two main panels. The panel on the left displays all available shelves and recipes in an expandable tree widget. An expander widget to the left of a shelf name indicates that the shelf contains a non-zero number of recipes. All uncategorized recipes are managed in a special *Uncategorized* shelf.

The panel on the right is the display area for the recipe and shelf specification canvas. If a shelf is selected in the shelves and recipes tree, the shelf details are

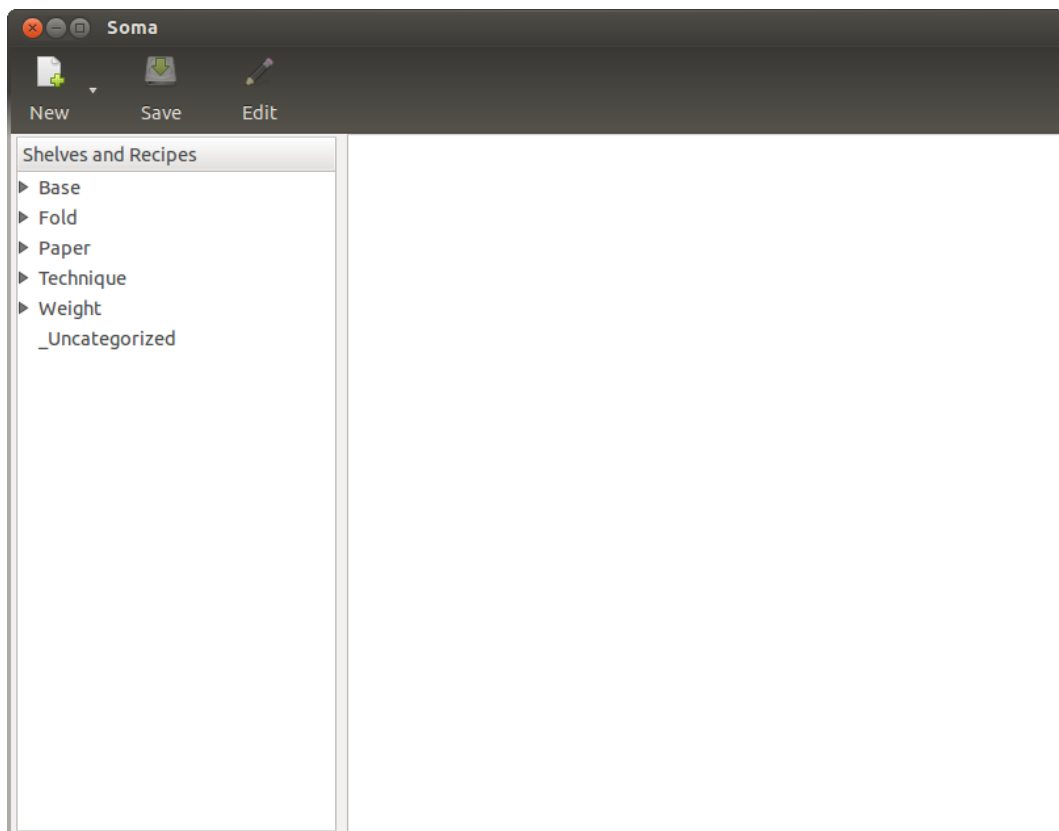


Figure 7.9: Opening screen of Soma.

displayed in the specification canvas. Additionally, the specification canvas can also be used to create a new recipe or edit an existing recipe.

Operations such as creating a new recipe or specification, or saving and editing an existing recipe or specification are initiated from the App toolbar (Figure 7.10).

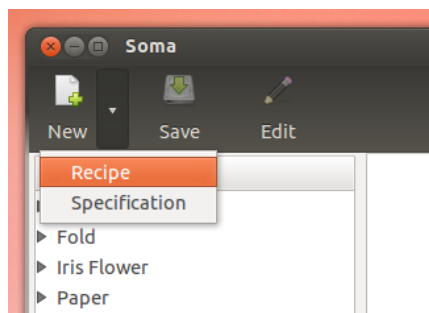


Figure 7.10: Creating a new recipe.

A recipe's data structure (illustrated in Figure 7.8) is initialized using an interactive dialog where the user provides information about the firmness of the recipe. While the user can enter the name and descriptive values for the recipe (Figure 7.11), the UUID for the recipe is auto-generated by the tool. The generated UUID is a version 4 UUID, thus ensuring a significantly unique identifier for all recipes managed by the tool.

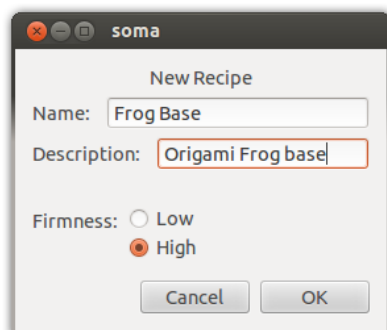


Figure 7.11: Filling in the details for a new recipe. The recipe UUID is auto-generated.

All newly created recipes are added to the *_Uncategorized* shelf. This shelf is specific to the Soma implementation and houses recipes which are not currently assigned to any domain-specific categorizations.

While each recipe is characterized by the four attributes (Recipe ID, Name, Description and Firmness) illustrated in Figure 7.8, the specific nature of each recipe determines its level of detail encoded in its definition.

56d0ac84-0525-4234-a8ec-0722c1a7e66d
Name : Flat Technique Description : Flat technique of folding Firmness : HIGH Detail : The flat technique for folding paper in origami. The paper is kept on a flat horizontal surface and folded.

Figure 7.12: A recipe for the Flat technique.

For instance, one particular recipe for the *Flat Folding Technique* may be as simple as the one-line description illustrated in Figure 7.12, while a recipe for the *Valley Fold* (Figure 7.13) may reference external data sources.

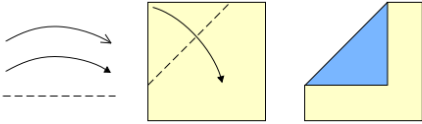
ac72b4a5-0f74-4704-8a8a-ea8d3d8884e3
Name : Valley Fold Description : Valley fold in origami Firmness : HIGH Detail : See https://en.wikipedia.org/wiki/File:Origami_symbol_valley_fold.svg


Figure 7.13: A recipe for Valley fold.

Thus, the exact nature and detail of recipe encoding depends not only on the nature of the domain to which the recipe belongs, but also on the level of abstraction, i.e., concept, model or implementation.

Creating a Shelf

The creation of a new shelf in Soma is similar to the process for a recipe.

As seen from Figure 7.8, every shelf is a recipe with predetermined presets. The only difference is the absence of user-input for firmness — a new shelf is implicitly of low firmness in Soma (Figure 7.14).

An external shelf for future use is created by simply creating a new shelf and adding recipes to it. Recipes can be added to a problem-domain shelf by dragging them from the available shelves and recipes. Figure 7.15 illustrates the result after

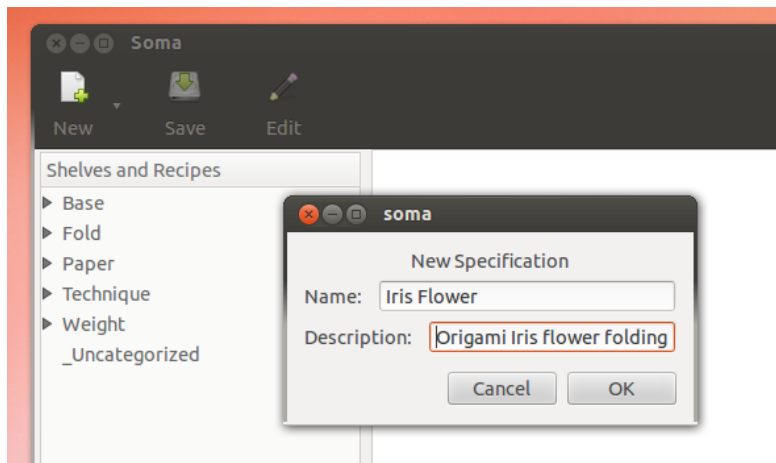


Figure 7.14: Entering the details of a new specification.

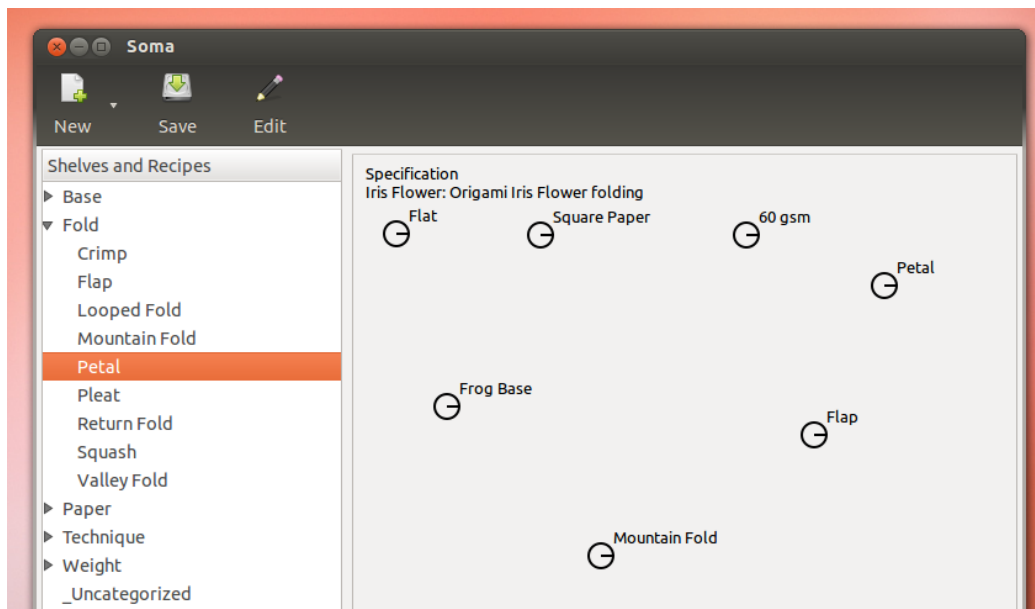


Figure 7.15: Recipes in the Problem-domain Shelf view.

concern refinement for the problem–domain shelf of the origami iris flower folding workflow.

7.4.2 Soma Sprint 2

Features in the Soma Logical Layer and the Soma Visual Layer, and the formulation and specification of context relations between recipes in the solution specification are implemented in this sprint.

Solution Specification Formulation

Right-clicking a prototype in the specification canvas presents a menu for setting the recipe state (Figure 7.16).

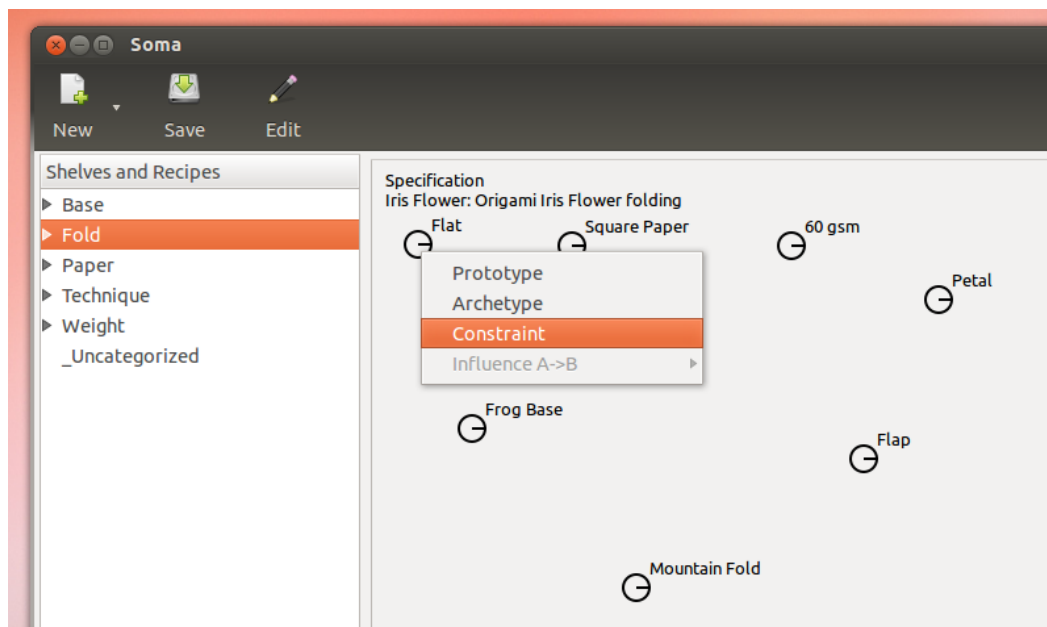


Figure 7.16: Assigning recipe state.

Depending on problem analysis, a prototype can be configured as either an archetype or a constraint.

Once the constraint is selected, the symbol displayed for the recipe (circle) changes to a rhombus to show that the recipe has been configured as a constraint. For instance, the *Flat*, *Square Paper* and *60gsm* recipes have been identified as constraints in Figure 7.17.

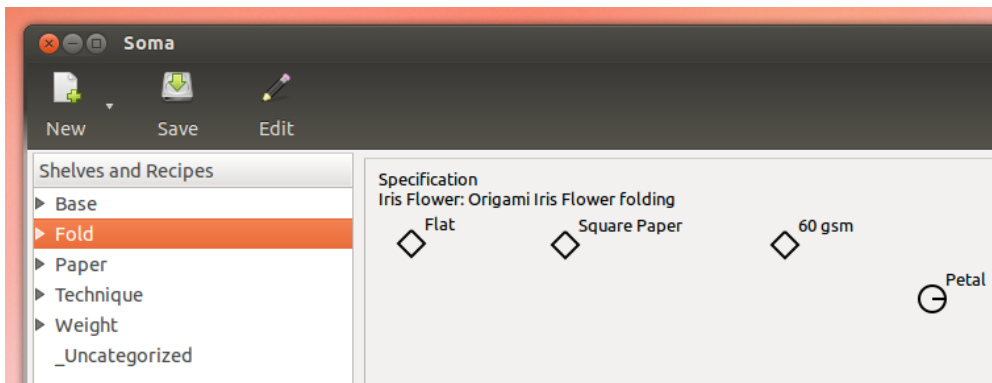


Figure 7.17: Constraints for the origami iris flower specification. The *Frog Base*, *Mountain Fold* and *Flap* recipes are not shown in this figure.

Specifying Context Relations

The contextual relationship between two recipes is defined by first selecting them and then assigning a value for the influence of the first recipe on the second. As described in Section 3.16, for any two recipes A and B, the context relation is defined as $C_{AB}(I_{AB}, F_A)$. As illustrated in Figure 7.18, in the Origami problem of folding an Iris flower, a context relation between the *Square Paper* constraint and the *Frog Base* recipe can be defined by selecting these two recipes. Upon selection, the color of the first recipe (Recipe A) selected for the context relation changes to red. Similarly, the color of the second recipe in this relation (Recipe B) changes to blue when it is selected for specifying the context relationship.

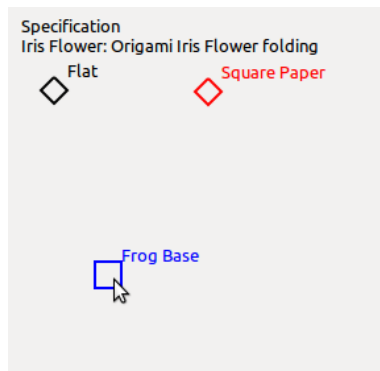


Figure 7.18: Selecting recipes for context refinement.

Next, right-clicking Recipe B (*Frog Base* recipe) displays the popup menu for selecting the influence exerted by Recipe A (*Square Paper* recipe). This is illustrated in Figure 7.19.

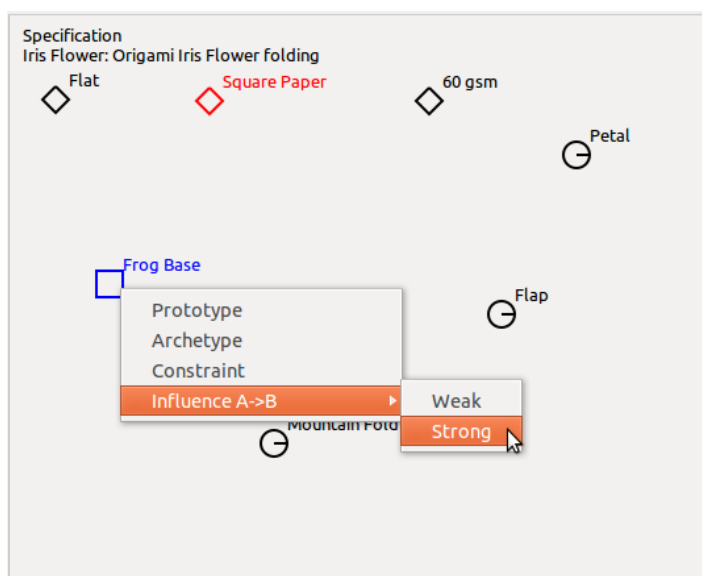


Figure 7.19: Assigning recipe influence.

Saving the Shelf

After determining additional context relations between other recipes which are selected for the problem specification, the solution shelf is saved by selecting the Save option in the toolbar. Once the solution specification is saved, it is added to the list of available shelves for further use. As seen in Figure 7.20, a new shelf for the Iris flower specification is added to the list of shelves and recipes in the left panel.

This newly created solution shelf recipe for folding an iris flower can now be refined further or reused in a new problem specification as required. The use of Soma in the specification of the iris flower folding workflow (Chapter 4) described in this chapter is also illustrated in a short demonstration [Chemboli, 2012a].

7.5 SUMMARY AND CONCLUSION

Even though the OAR processes may be carried out manually, there is a need for systematic tooling support for recipe and shelf management using the processes of concern and context refinement. Such support is not only desirable but also necessary to apply OAR to problems involving large number of concerns and recipes. As a first step in this direction, the design and implementation of Soma, a prototype tool for OAR, is presented in this chapter. The following requirements were identified for the tool:

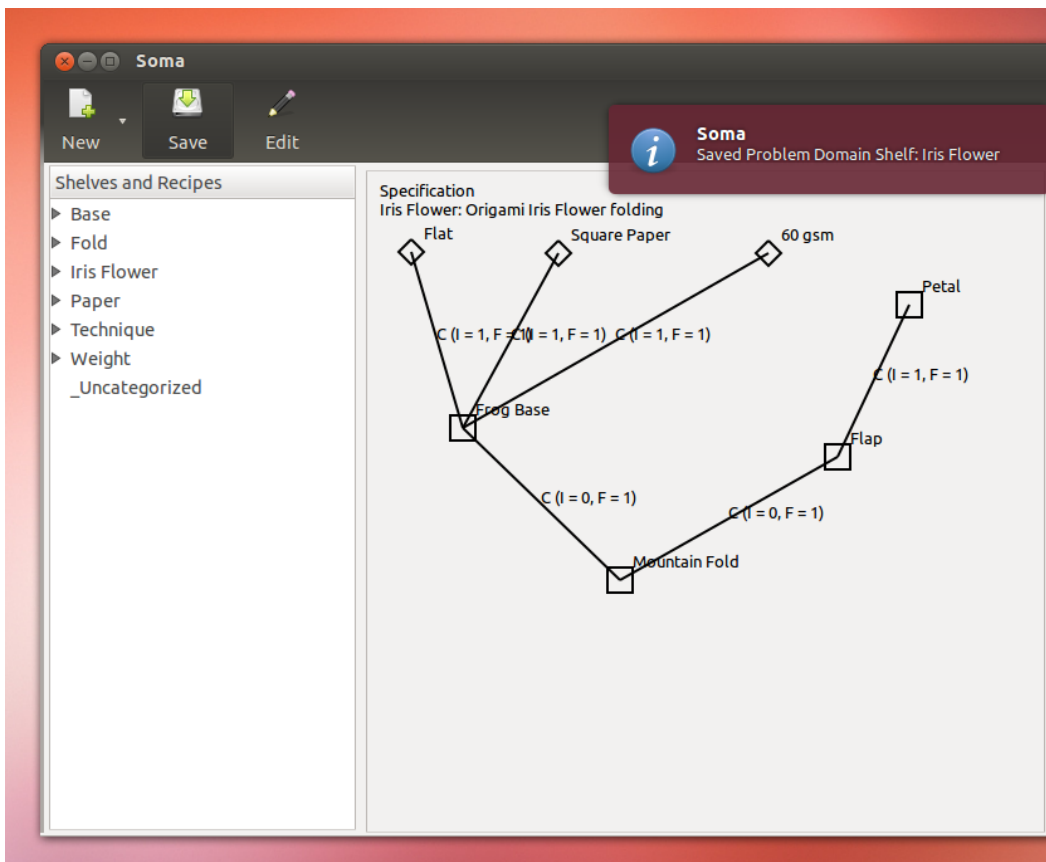


Figure 7.20: Saving the solution specification.

1. Unambiguous management of ukes and uke groups in recipes by assigning UUIDs.
2. Categorizing and managing recipes in shelves.
3. Support for defining and populating a problem–domain shelf using concern refinement.
4. Solution shelf specification by context refinement.
5. Visual representation of recipe and shelf management.

The development of the reference tool implementation is carried out using the Python–GTK+3 development stack for the Ubuntu 12.04 LTS platform. A SQLite database is employed in this implementation.

The implementation described in this chapter is simple and rudimentary; it is meant to highlight the essential features that are required of a tool for OAR. The addition of features such as drag–and–drop editing of shelves, handling directionality of context relations, tagging and categorization support for incorporating provenance data for recipes and adding support for translation engine recipes for implementation–level workflows is desirable for handling large–scale workflows with Soma. While implementing these features in Soma is beyond the scope of the present thesis, they constitute important future directions of research for robust tool development for OAR.

8

SUMMARY AND CONCLUSIONS

Now I begin to think my reputation, such as it is, will suffer shipwreck if I am so candid. 'Omne ignotum pro magnifico.'

Everything becomes commonplace by explanation.

The Adventures of Sherlock Holmes: The Red-Headed League [1985]

8.1	Summary of Contribution	142
8.1.1	An Enhanced Definition of Scientific Workflow	142
8.1.2	Omnispective Analysis and Reasoning	142
8.2	Limitations of Contribution	145
8.3	Related Work	147
8.4	Viewing Enterprise Architecture through OAR	148
8.4.1	Enterprise Architecture and Architecture Frameworks	149
8.4.2	Applying OAR — Architecture Views as Workflows	152
8.5	Directions for Future Work	155
8.5.1	Managing Fractional Values for Firmness and Influence in Recipe Context	155
8.5.2	Tool Support	155
8.5.3	Moodle OAR Plugin	156
8.5.4	Application to Complex Systems	156
8.5.5	OAR as Science of Workflows	156

The aim of this thesis is to *formulate, develop and demonstrate an epistemic framework for managing intellectual concerns in scientific workflows*. This aim has been achieved through the conception, formulation, development and demonstration of *Omnispective Analysis and Reasoning (OAR)*.

This chapter is organized as follows. A discussion of the main contributions made in the thesis is presented in [Section 8.1](#). The scope and limitations of the contribution are considered in [Section 8.2](#). A discussion of similar and related work is included in [Section 8.3](#). This is followed by a brief discussion on viewing enterprise architecture through OAR in [Section 8.4](#). Finally, [Section 8.5](#) presents recommendations for future directions of research.

8.1 SUMMARY OF CONTRIBUTION

The research reported in this thesis has resulted in the following contributions.

8.1.1 An Enhanced Definition of Scientific Workflow

It has been shown in chapters 2 and 3 that the current practice of scientific workflows is restricted to the programming of steps for designing and implementing experimental processes and data management techniques. The term scientific workflow in current parlance has been established by ad hoc usage without formal definition.

Since scientific workflows deal with all the facets of a scientific project, the Scientific Method is applied in this thesis to arrive at a well-rounded and enhanced definition of scientific workflow beyond the currently understood scope of *computer flowcharting*:

A scientific workflow is a representation of any logical, systematic and repeatable inquiry, investigation and corresponding set of actions.

This is a *Definition by Relationship* which extends the scope of ordered analysis and investigation to any problem scenario. In addition to incorporating the traditional view of scientific workflows, any workflow that has been specified in terms of a well-formed structure consisting of *Foundation, Theory* and *Methodology* is covered by this definition.

This definition brings under the purview of scientific workflow management even those problems which may not be conventionally described as *scientific activity*, but may yet be analyzed, modeled and orchestrated by the scientific method. Any problem that can be specified in terms of a *basis*, parameters, processes and interactions, can be handled by a scientific workflow. Thus, according to the enhanced definition formulated in this thesis, the concept of a scientific workflow need not be restricted to *structured steps of laboratory processes*.

8.1.2 Omniscpective Analysis and Reasoning

It has been brought out in [Chapter 2](#) that the management of intellectual concerns in current scientific workflow practice is unsatisfactory. The following factors were identified:

1. Focus on low-level implementation details.

2. Inadequate context support and management.
3. Depending on the scale of the problem, the task of developing shared semantics across disciplines can become intractable and unmanageable.
4. Lack of support for verification and validation of the underlying science of the workflow.

These observations led to the conjecture that the management of intellectual concerns can be improved by:

1. Providing a layer of abstraction to lift focus from low-level implementation details.
2. Adding context as a parameter of scientific workflow components and participating processes.
3. Introducing localized ontologies applicable to a particular instance of a problem situation; and
4. Providing an abstraction to capture and organize intellectual concerns in the problem domain and map them to the workflow specification and execution semantics.

Based on this conjecture, the ideas which form the foundations of *Omnispective Analysis and Reasoning* were conceived, resulting in the formulation and development of the *OAR framework* which is presented in [Chapter 3](#).

The concepts developed in OAR are:

1. *Omnispection, Analysis and Reasoning*: Omnispection provides an overall appraisal of a system. Analysis entails a close examination of all the recognized entities contained therein. Discovery of the relationships and interactions between the entities and also with their environment is facilitated by Reasoning.
2. *CMI Hierarchy*: A concern or entity present in a problem domain may be abstracted at three levels of conceptualization — the Concept, Model and Implementation levels.
3. *Unit Knowledge Entities (ukes)*: Application of OAR results in assembling all the relevant information of the inquiry domain abstracted into Unit Knowledge Entities (ukes) at the *Concept, Model* or *Implementation* levels. This clearly encapsulates all the relevant interrelationships and forms the overall repertory from which the scope of the problem can be defined and a solution formulated.

4. *Context of a recipe*: Context in OAR specifies the scientific validity of a uke and its relevance to the problem under study. Uke context is a function of the attributes *Firmness* and *Influence*. Firmness is based on the validation of the recipe with reference to the foundation and theory of the science of the problem. The relevance of a recipe to the problem and solution specification is denoted by Influence.
5. *Prototypes, Archetypes and Constraints*: Recipes are groups of ukes formed for convenience. Depending on the degree of Influence and Firmness in relation to the problem under study, recipes are considered as *Prototypes*, *Archetypes* or *Constraints*. A prototype is a recipe without any consideration of its applicability; archetypes are recipes that are exemplars or best practice; a constraint is a special instance of an archetype which imposes strict criteria in the problem domain.
6. *Concern Refinement*: The OAR process of concern refinement is the identification of recipes which are relevant to the problem under consideration.
7. *Context Refinement*: Context refinement is the process of determining the relevance of recipes to a problem domain and developing a mapping of the contextual relationship between recipes within a solution specification. The processes of concern and context refinement together provide the *epistemological basis* that ensures verification, validation and mapping of the relevant recipes to the workflow.
8. *Managing recipes in shelves*: Recipes are managed in the OAR framework using shelves — simple unordered collections. These are categorized as *External*, *Problem-domain* and *Solution* shelves. External shelves hold all the known prototypes from different domains of interest in the analysis of the problem. A problem-domain shelf is populated with recipes selected from external shelves that satisfy given criteria in the problem under consideration. The solution shelf is an interconnected specification of archetypes in the problem domain subject to identified constraints.
9. *Localized Ontologies*: Shelf management facilitates localized ontologies in the OAR framework. These are formed by considering *satisficing* knowledge for the problem domain, thus relaxing the requirement of exhaustiveness in ontology development. If further analysis of the problem situation reveals additional interacting concerns, the localized ontologies may be further extended and refined.

The examples presented in chapters 4 to 6 demonstrate the applicability and potential of the OAR framework. The Origami workflow for Iris flower folding in

[Chapter 4](#) clearly illustrates the processes of OAR — verification and validation of recipes, concern and context refinement as well as the management of intellectual concerns by lifting focus from low-level implementation details.

Chapters [5](#) and [6](#) demonstrate the application of OAR to tackle generic workflow problems that can also be considered as systematic scientific activities. The application of OAR for contextualizing course design demonstrates the use of OAR in capturing the rationale of design decisions in a course and mapping them to desired outcomes. The application of OAR for the analysis, understanding and management of complex systems is described in [Chapter 6](#). Using the OAR framework, large systems can be analyzed and managed as an aggregation of localized ontologies with explicit specification of mutual interactions and influence. The omniscpective outlook, taking into account the details of individual subsystems and their interactions, facilitates verification and validation of the system with insight as well as management of the system development lifecycle. Localized ontologies, and fixing the control context, enable the exposure of side-effects and behavior that may emerge due to the choice of any particular solution specification.

A simple and illustrative implementation of Soma, a prototype tool for OAR, is presented in [Chapter 7](#). The current iteration of Soma provides support for managing and visualizing recipes and external, problem-domain and solution shelves. Though rudimentary in features, Soma is intended to be a first step in the development of a reference tool implementation for the OAR framework.

8.2 LIMITATIONS OF CONTRIBUTION

The processes and concepts used in the OAR framework are based on the ideas of Analytical Philosophy, and hence their limitation is only the limitation of human thought processes to-date. The OAR framework as such, does not suffer any limitations. The open and evolving nature of science is embodied in the OAR framework by adhering to the Domain of Science Model (DoSM) and utilizing the OODA refinement process. Hence, the framework may always be tuned to meet the task of analyzing a problem. Any limitations in demonstrating the applicability of OAR spring from the inherent limitations in the applications considered.

The specification of context within the OAR framework is currently limited to a simplified approximation of four discrete sets of influence and firmness values. Fractional values of context indicate vagueness and gaps in knowledge in the problem domain. Additional research is desirable to devise formalism for specifying and utilizing fractional values of influence and firmness.

Ascertaining how well OAR handles the management of intellectual concerns in comparison to current scientific workflow management methodologies cannot be reasonably addressed at this juncture because there is no standard for comparison. In current workflow practice, there is no provision for identifying, capturing, verifying and validating intellectual concerns of the research domain, nor there is any deliberate focus on ensuring and verifying that the correct scientific workflow is modeled and implemented for the problem situation. As discussed in chapters 2 and 3, current approaches mainly address the logistics of workflow implementation and orchestration, and any form of validation is normally carried out outside the workflow as an independent and separate activity.

The choice of simple and concise applications to illustrate the concepts and application of the OAR framework is not a limitation imposed by OAR. The examples are deliberately chosen to be pithy and lucid enough to satisfy the aim of *illustrating and explaining* OAR. The purpose is not to make a ‘discovery’ in either origami or course design or complex systems, but to provide an adequate illustration and demonstration of the OAR framework. Tackling a particular problem in this exercise is incidental.

Hence, raising objections like — origami is not a scientific workflow, the course design illustrated may not necessarily be the most desirable one that can be produced, a ‘complex system’ should be ‘more complex’ than the Ubuntu 12.04 LTS ecosystem — is not relevant here. Such objections, as Peirce [1877] argues, would only stem from *tenacity in fixing belief*.

The applications presented in chapters 4 to 6 are manageable examples and serve to demonstrate the main features and applicability of the OAR framework. Large-scale evaluation of Omnispersive Analysis and Reasoning would require engagement with full-fledged scientific research projects where the OAR framework guides the planning and execution of the research. The application of OAR to research problems of significant scale involves considerable effort in proportion to the magnitude of the problem. This, in turn, requires comprehensive end-to-end tool support for better manageability — the development of such tooling is beyond the scope of this thesis. While the reference tool implementation for OAR (Soma) presented in Chapter 7 enables simple management and visualization of recipes and shelves, it currently lacks features such as provenance support and recipes for translation engines for handling large-scale workflows.

8.3 RELATED WORK

The issues affecting the management of intellectual concerns in scientific workflows have been discussed in chapters 2 and 3 of this thesis. It was also pointed out that in so far as it could be ascertained, there are no indications in literature of any significant approaches attempting to address these issues. So, finding ‘related work’ can only be based on seeking a similarity of thinking in other research contexts. Such similarities may not be readily visible.

Integration and Implementation Sciences (I2S) [Bammer, 2006] is a new initiative that attempts to bring together experts and scientific information from different disciplines and facilitate interaction and dialog through a common hub, for the purpose of creating a pool of interdisciplinary knowledge. Here the parallel one can discern is with the idea of Omnispection since the I2S initiative attempts to catalog domain-wise information. The focus of I2S is mainly to provide an information base that may be useful across multidisciplinary projects. On the other hand, the OAR framework represents the result of focused research — to analyze a problem, formulate a workflow specification, and determine means of workflow implementation.

Some instances can be cited which have ideas somewhat similar to the concept of localized ontologies formulated in OAR. Unified Structured Inventive Thinking (USIT) [Sickafus, 1997] is a structured, problem-solving methodology primarily used in the automotive industry for obtaining solutions to engineering problems by focusing on a limited set of available information resources, and ‘inventing’ approaches to remove the difficulties encountered. Although USIT does not care to dwell upon theoretical considerations, the functioning of this methodology displays similarity to the operation of Localized Ontologies realized in the OAR framework through the use of shelf management. Aspect-Oriented Thinking (AOT) [Flint, 2006] is an approach to analyze multidisciplinary problem situations. The different disciplines involved in the problem are separated into autonomous domains, each domain corresponding to one particular discipline. Domain knowledge is captured in a domain model and a set of such models spanning a variety of disciplines is invoked to create the system. Partitioned Iterative Development [Sessions, 2006] is a process suggested for rapid development of system architectures. Partitioning a large system into parts based on functions or activities and prioritizing development of the partitions based on lower cost and higher visibility of the outcomes is expected to increase the chances of rapid development. The set of domain models of AOT and the partitions in the partitioned iterative development may be considered as concepts somewhat similar to localized ontologies in OAR.

In OAR the term context stands for the well-formedness of a uke (designated as firmness F) and its influence on another uke (designated as influence I) and is denoted as $C(F, I)$. Here, context as a parameter has been introduced and employed in a manner analogous to a thermodynamic parameter, making it useful for identifying archetypes and constraints and for unambiguously realizing problem and solution specifications of a workflow. A model of ‘context’ which has been used in analyzing relationships in a problem [Alshaikh and Boughton, 2009] has been mentioned in [Chapter 2](#). This model of context uses two conceptual entities ‘perception’ and ‘influence’; if the connection between two objects is ‘obvious’, the context is termed as ‘explicit’ and in the absence of such obvious connection it is termed ‘implicit’. Influence is considered to be a ‘force’ graded from a tangible feature like ‘fit’ to notional features like ‘passion’ and ‘culture’. Alshaikh and Boughton [2009] observe that their formulation when applied to a problem, may result in some context relations that seem to be ‘incorrect’ or ‘awkward’, and therefore would require further development and refinement.

8.4 VIEWING ENTERPRISE ARCHITECTURE THROUGH OAR

The growing importance of architectures and architecture frameworks in the design and management of large software-intensive systems, has been well-recognized for over two decades [Garlan and Shaw, 1993]. Nowadays it is trendy to mention Architectures and Architecture Frameworks like Department of Defense Architecture Framework (DoDAF) [DoD CIO, 2010] while discussing frameworks of any kind. This is particularly so in view of the visibility of such architecture frameworks and the involvement of several reputed agencies, tool developers and vendors in developing and advocating their use and usefulness.

It has been suggested [Boughton, C. (May 2012), private communication] that it would be interesting to consider how enterprise architecture frameworks stand when viewed with reference to the processes of OAR.

At first sight, it may appear extraneous to include a discussion of enterprise architecture framework in the context of presenting OAR which is an epistemic and theoretical framework. However, one can contend that there is a certain relevance based on two considerations:

1. The tradition of software engineering research has been adaptive inclusion of ideas and concepts from a variety of domains and this would encourage foraging into other domains.

2. In OAR, the concept of a workflow has been generalized to include any logical and systematic inquiry. The views produced by means of an architecture framework are based on a logical and systematic inquiry directed at the architecture description.

A brief overview of architecture frameworks is presented in this section. As a representative case, the salient features of DoDAF are given. The similarity of an architecture view to a service workflow is pointed out. The process of generating architecture views using OAR is outlined.

8.4.1 Enterprise Architecture and Architecture Frameworks

The ISO/IEC/IEEE/42010 standard [ISO/IEC/IEEE, 2011] defines architecture and related terms. An architecture according to the standard is defined as:

Fundamental concepts or properties of a system in its environment embodied in its elements, relationships and in the principles of its design and evolution.

According to this definition, an architecture is a set of concepts pertaining to the essential features of a system, or it may stand for a set of properties of the system as perceived by the designer (referred to as an Architect). All the artifacts used to express and document the architecture are considered as comprising an Architecture Description (AD).

An architecture has the following key features:

1. An architecture exists in the context of its environment and consists of the essential concepts and/or perception of the system represented by it.
2. The features of an architecture are archived in the form of an architecture description. The AD is required to indicate clearly who are the individual stakeholders and how their needs and concerns are fulfilled.
3. Addressing stakeholder concerns is realized through a number of Architecture Views of the system. Each view is designed in such a way as to clearly present an identified and pre-determined set of stakeholder concerns.
4. The rules and processes for generating the views are prescribed by an Architecture Viewpoint (AVP). These rules and processes are to be explicit and unambiguous in producing the view. Each view, by design, has to be specific to address the concerns of a particular stakeholder.

Since Architectures can be as diverse as the systems they represent, with different systems having vastly different sets of concerns and stakeholders, the standard does not specify how many views or what kind of views are to be incorporated; it only insists that all stakeholders should be identified and their concerns presented as views.

An Architecture Framework (AF) embodies all the conventions, principles and practices that are formulated for the purpose of describing the architecture specific to a particular domain of application.

All the details of the features of the architecture, architecture description, views and viewpoints are captured and documented in the form of a *Conceptual Model* or meta-model of the Architecture. The meta-model is expected to articulate explicitly all the key concepts and terms for providing the Architecture Description. Additionally, it should encompass all the features of the architecture framework, the Architecture Design Language (ADL) and the viewpoints. The AF must clearly spell out which are the concerns, the stakeholders who have these concerns and the viewpoints that capture these concerns. Consistency between the different views generated is assured by including in the architecture framework suitable rules integrating the viewpoints. Thus a system is represented by an Architecture, the details of which are given by the Architecture Descriptions that are embodied in a Framework consisting of ViewPoints that can be utilized to produce Views for different stakeholders. The purpose of the Views is to enable the stakeholders to carry out *necessary action* in the management of the system.

In a fast-growing field like Software Engineering, innovative applications do not wait for standards to be formulated; it is rather the standards would try to categorize and describe the innovations and record the best practices. A number of Enterprise Architectures have come into vogue with a spread in the meaning of the term Architecture with confusion (a lament indicated in the Standard) between *abstract ideas* and the *artifacts* used for describing the ideas.

Several enterprise architecture frameworks, proprietary as well as open-source, are being used in various organizations and enterprises. Some of these are developed by consortia of industry and research organizations and some by defense and other governmental agencies.

A comparative analysis of six architecture frameworks in terms of how the goals, inputs and outcomes have been addressed in the frameworks is presented by Tang, Han, and Chen [2004]. They find that all the frameworks examined support the purpose of software architecture development to various degrees and the frameworks TOGAF, DoDAF and FEAF additionally address issues of architecture planning, evolution and system interoperability. They point out one main deficiency — the level of detail required in an architectural model is generally

not specified and the architecture rationale is not a mandatory part of the model and consequently the architecture models cannot be verified or traced.

A concise account of the various processes involved, the status and challenges of enterprise architecture is given by Op't Land et al. [2009]. They describe enterprise architecture as an “instrument for informed governance” which is still “in early stages of development” with no scientifically documented evidence of success in application. It is pointed out that the practice of enterprise architecture has yet to demonstrate its effectiveness “to aid organizations in solving their transformation problems in a repeatable and predictable fashion,” and the need for fundamental research to clarify several aspects of the field is advocated.

As an example of a typical architectural framework, the salient features of DoDAF are reproduced below [DoD CIO, 2010; IEEE, 2012]:

ID DoDAF

NAME US Department of Defense Architecture Framework

PURPOSE To enable the development of architectures to facilitate the ability of Department of Defense (DoD) managers at all levels to make key decisions more effectively through organized information sharing across the Department Joint Capability Areas (JCAS), Mission Component and program boundaries.

SCOPE US DoD

VIEWPOINTS All; Capability; Data and Information; Operational; Project; Services; Standards; Systems

DoDAF, essentially, is specification of a standard for Enterprise Architecture Framework in the domain of US Department of Defense. This consists of a framework and a conceptual model to enable development of a number of architecture views for the use of managers at various levels. The views are meant to assist management and decision-making functions. How the data has to be exchanged is prescribed by the DoDAF Meta-Model (DM2). The standard and the meta-model contain details and guidelines for tool designers and vendors to follow in developing and producing tool support.

The configuration — structure and details of content — of DoDAF has evolved over the past twenty years as a continuing effort of the Department. The DoDAF viewpoint definitions, model specifications, glossary, and DM2 meta-model are still undergoing improvements and clarifications with the participation of various stakeholders [DoD CIO, Architecture & Infrastructural Directorate, 2012]. Tools by various vendors are at different levels of development and readiness, and the

ongoing “Configuration Management Process” is scheduled to be completed by the end of 2012.

It is interesting to note that the evolution and development of DoDAF spans the development timeframe of enterprise architecture and architecture framework evolution. Hence the documentation of the various stages of DoDAF development is a good source for the historian to trace the evolution and development of this technology.

8.4.2 Applying OAR — Architecture Views as Workflows

As pointed out in [Chapter 2](#), a marked distinction exists between the conventional view of business workflows and scientific workflows concerning the way the workflow may evolve and change during execution and the agent responsible for specifying the workflow process. A business process specialist or administrator is responsible for setting up a business workflow. Once formulated, the workflow is used in a routine manner with minimal changes over time. In architecture frameworks, the viewpoints are predefined to produce the views as outcomes. Hence, these fall under the category of business workflows. Alternately, one may consider these as service oriented workflows.

Another way of looking at architecture frameworks is that they are database management systems with report generating tools incorporated to produce reports (views) having a predetermined structure and content with the constraint that all of the several views generated are consistent with each other and with the contents of the database (which in this case is the architecture description). Epistemological considerations do not figure in these frameworks.

The process of enterprise architecture development consists of four steps — data collection, preliminary view generation, review and revision of the views and publishing the views [CIO Council, 2001]. The framework requires support of relevant tools at various stages of this process. Building an architecture is steeped in a variety of implementation details. All the nouns, verbs and connective phrases of architecture viewpoints, procedures and outcomes are geared to low-level procedural constructs. Since there can be several flavors of enterprise architectures, they will be considered in abstract terms to keep generality for the purpose of examining them in terms of OAR.

Collecting data sufficient to provide adequate and clear description of the enterprise is akin to omnispersion. Generating preliminary views and refining them is like formulating models of the process and publishing the views is the implementation providing the outcomes. This, in short, is a concept-level view of enterprise architecture.

Since the procedures for data validation and the rationale of the models used are not mandated in the framework, validation of data and verification of the models depend on the strategies, policies, present state and vision of the enterprise (which are incorporated in the architecture meta-model). Consequently, validation of data and verification of models is not an open feature and no epistemological considerations are specified in these frameworks.

Stakeholder concerns are addressed by providing views that are designed to present the concerns considered relevant at a level of detail appropriate to the role of the stakeholder. The rules, procedures and details of the information to be presented are specified in a viewpoint. Details of stakeholders and their concerns, viewpoints and views, rules and tools needed for implementation fall within the purview of the architecture framework and are defined and specified in the architecture meta-model.

In OAR, the analysis, design and operation of a system is considered as a workflow formulation and solution specification. All identified entities in the system — which include details of component systems and their interactions, conventions, rules, policies, procedures, software implementations and tools — which constitute the architecture description are the recipes in external shelves.

To obtain a viewpoint appropriate for a given stakeholder, the OAR process of concern refinement is carried out as per the prescription of the meta-model to select relevant recipes into the problem-domain shelf. Context refinement gives the viewpoint in the solution shelf. Viewpoints so generated for different stakeholders are considered as recipes and held in a viewpoints external shelf for future use.

The workflow for generating the views is analogous.

The problem-domain shelf contains the relevant viewpoint, tools and constraints as prescribed by the meta-model. A conceptual representation of the OAR process for realizing the view is shown in [Figure 8.1](#). The views generated are collected in a views shelf for reuse.

Any changes in the status of the concerns or specifications in the meta-model are incorporated by adding new recipes or modifying existing recipes. Such changes may be recorded in the representation of the recipes to indicate their evolution.

Examining the context of recipes (contextualizing the recipes with respect to the outcomes) exposes the concerns that might have become irrelevant or relevant but not adequately enunciated. The former are considered for retirement or replacement while the latter represent 'gaps' in the architecture description and require further study.

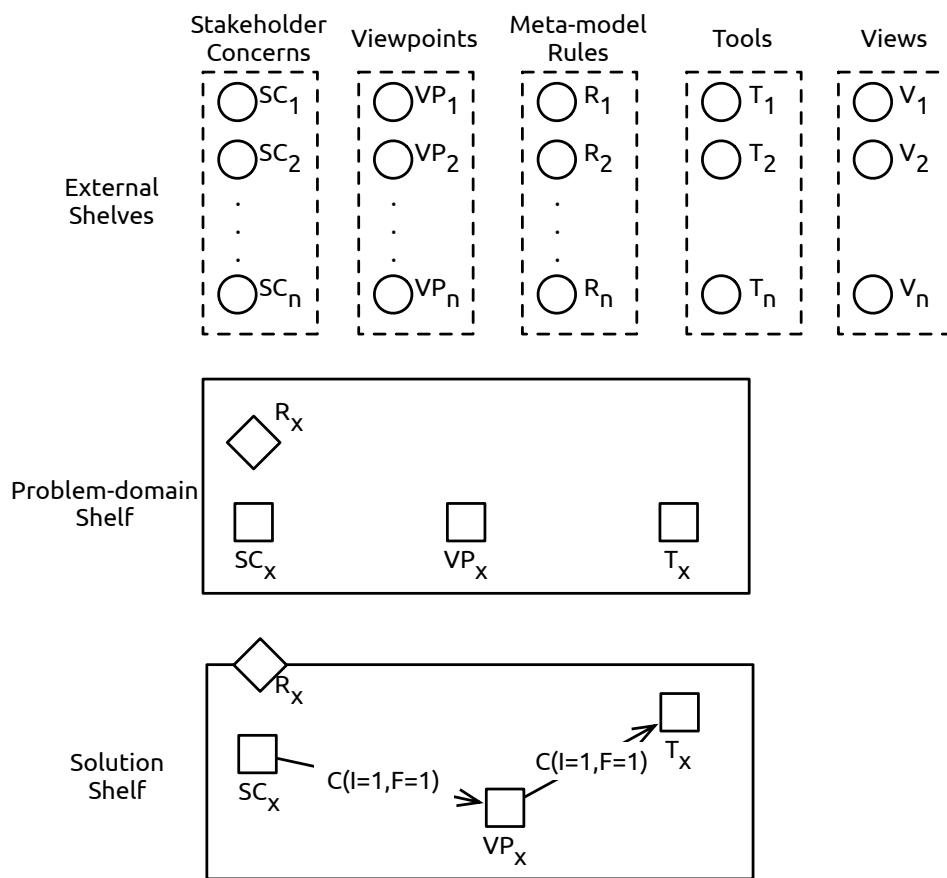


Figure 8.1: OAR process for generating architecture views.

The OAR processes of concern and context refinement ensure that the views generated are consistent with each other as well as with the overall system description provided by the recipes in the external shelves.

The application of OAR can be carried out at different levels of granularity, and facilitates integrated management of a system with reference to the criteria incorporated in the architecture framework. OAR does not impose any scalability restrictions, any issues of scalability and interoperability only arise from the technology and tools employed for implementing the architecture.

8.5 DIRECTIONS FOR FUTURE WORK

Some activities for future research and development of the OAR framework are given in this section.

8.5.1 Managing Fractional Values for Firmness and Influence in Recipe Context

The existence of fractional values for firmness and influence implies that exact solution specifications may not be possible with the available set of recipes. This reflects vagueness and lack of knowledge in the understanding of the problem situation. As discussed in [Section 3.17](#), recipe context in OAR is currently represented by four discrete context tags as a practical approximation. “Managing unknowns is just as important as making maximum use of what is known when responding to real world problems” [Bammer and Smithson, 2008]. Thus it would be useful to evolve a formalism for managing fractional values for firmness and influence and using them to characterize vagueness and associated risks that may evolve from the use of ‘imprecise’ solution specifications.

8.5.2 Tool Support

Extending Soma by integrating existing scientific workflow management systems and methodologies as implementation level recipes in OAR solution specifications will provide support for translating the solution shelf to executable form. Additionally, it is desirable to develop end-to-end tool support to facilitate the application of OAR to large systems. Incorporating automated reasoning support for OAR tasks and processes, and providing support for rapidly bootstrapping external shelves with domain-specific knowledge will significantly contribute to the management of large projects.

8.5.3 Moodle OAR Plugin

Application of the OAR framework to contextualize course design has been presented in [Chapter 5](#). This could be further extended by the development of a workflow plugin for the Moodle LMS to facilitate the capture and representation of intellectual concerns and rationale in course design and implement them using Moodle tools and resources. Such a plugin would improve the connection between the goals and intent of a course and its delivery using the Moodle LMS.

8.5.4 Application to Complex Systems

The application of OAR to the analysis and understanding of a complex system has been considered in [Chapter 6](#). Further studies are necessary to fully explore the scope and extent of benefits of the application of OAR to the analysis and management of complex systems.

8.5.5 OAR as Science of Workflows

The need for the development of a science of workflows has been felt for quite some time in the workflow community [Deelman and Gil, 2006b]. The foundation, theory and methodology for a science of workflow management have been proposed, developed and designed as the OAR framework in this thesis. The applicability of the framework has been demonstrated by examples. In other words, the contribution in this thesis corresponds to an instance of the Domain of Science Model (DoSM) for workflow management. Thus, if augmented with reasoning systems, implementation engines and knowledge bases for different domains, the OAR framework has great potential for evolving into a well-formed Science of Workflows.

BIBLIOGRAPHY

- Abbott, R. (2006). "Open at the top; open at the bottom; and continually (but slowly) evolving." In: *2006 IEEE/SMC International Conference on System of Systems Engineering*. IEEE. DOI: [10.1109/SYBOSE.2006.1652271](https://doi.org/10.1109/SYBOSE.2006.1652271) (cit. on p. 109).
- Adelsberger, H., Körner, F., and Pawlowski, J. (1999). "Continuous improvement of workflow models using an explorative learning environment." In: *Proceedings of World Conference on Educational Media EdMedia 1999*, pp. 506–511 (cit. on p. 18).
- Akarsu, E., Fox, G., Furmanski, W., and Haupt, T. (1998). "WebFlow: high-level programming environment and visual authoring toolkit for high performance distributed computing." In: *Proceedings of the 1998 ACM/IEEE conference on Supercomputing*. DOI: [10.1109/SC.1998.10046](https://doi.org/10.1109/SC.1998.10046) (cit. on pp. 15, 21).
- Alexander, C. (1964). *Notes on the Synthesis of Form*. Harvard University Press. ISBN: 978-0674627512 (cit. on pp. 27, 105).
- Alshaikh, Z. and Boughton, C. (2009). "The Context Dynamics Matrix (CDM): An Approach to Modeling Context." In: *Proceedings of the 16th Asia-Pacific Software Engineering Conference*, pp. 101–108. DOI: [10.1109/APSEC.2009.74](https://doi.org/10.1109/APSEC.2009.74) (cit. on pp. 27, 148).
- Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludäscher, B., and Mock, S. (2004). "Kepler: an extensible system for design and execution of scientific workflows." In: *Proceedings of the 16th International Conference on Scientific and Statistical Database Management*, pp. 423–424. DOI: [10.1109/SSDM.2004.1311241](https://doi.org/10.1109/SSDM.2004.1311241) (cit. on pp. 20–22, 46).
- Ardissono, L., Furnari, R., Goy, A., Petrone, G., and Segnan, M. (2007). "Context-aware workflow management." In: *Web Engineering*. Lecture Notes in Computer Science 4607. Ed. by Baresi, L., Fraternali, P., and Houben, G., 47–52. DOI: [10.1007/978-3-540-73597-7_4](https://doi.org/10.1007/978-3-540-73597-7_4) (cit. on p. 26).
- Arnold, V. (1986). *Catastrophe Theory*. 2nd ed. Springer-Verlag. ISBN: 3540161996 (cit. on p. 106).
- Baker, N., McClatchey, R., and Le Goff, J. (1997). "Scientific workflow management in a distributed production environment." In: *Proceedings of the First International*

- Enterprise Distributed Object Computing Workshop (EDOC 97)*, pp. 291–299. DOI: [10.1109/EDOC.1997.628370](https://doi.org/10.1109/EDOC.1997.628370) (cit. on pp. 15, 21, 61).
- Bammer, G. (2006). “Integration and Implementation Sciences: Building a New Specialisation.” In: *Complex science for a complex world: exploring human ecosystems with agents*. Ed. by Pascal, P. and Batten, D. ANU E Press, pp. 95–107. ISBN: 1920942386 (cit. on p. 147).
- Bammer, G. and Smithson, M. (2008). “Understanding uncertainty.” In: *Integration Insights 7*. URL: http://i2s.anu.edu.au/sites/default/files/integration-insights/integration-insight_7.pdf (cit. on p. 155).
- Barker, A. and Hemert, J. (2008). “Scientific Workflow: A Survey and Research Directions.” In: *Parallel Processing and Applied Mathematics*. Lecture Notes in Computer Science 4967. Ed. by Wyrzykowski, R., Dongarra, J., Karczewski, K., and Wasniewski, J., pp. 746–753. DOI: [10.1007/978-3-540-68111-3_78](https://doi.org/10.1007/978-3-540-68111-3_78) (cit. on p. 20).
- Barthelmess, P. and Wainer, J. (1995). “Workflow systems: a few definitions and a few suggestions.” In: *Proceedings of conference on Organizational computing systems*. Milpitas, California, United States: ACM, pp. 138–147. DOI: [10.1145/224019.224033](https://doi.org/10.1145/224019.224033) (cit. on pp. 12, 20).
- Berkley, C., Bowers, S., Jones, M., Ludäscher, B., Schildhauer, M., and Tao, J. (2005). “Incorporating semantics in scientific workflow authoring.” In: *Proceedings of the 17th international conference on Scientific and statistical database management*, pp. 75–78. ISBN: 1-88888-111-X (cit. on pp. 29, 59).
- Biggs, J. and Tang, C. (2007). *Teaching for Quality Learning at University: What the Student Does*. 3rd ed. Open University Press. ISBN: 9780335221264 (cit. on pp. 92, 94, 95, 97).
- Bloom, B. (1963). *Taxonomy of Educational Objectives: The Classification of Educational Goals: Handbook 1: Cognitive Domain*. New York: D. McKay (cit. on p. 100).
- Blumenthal, R. and Nutt, G. (1995). “Supporting unstructured workflow activities in the Bramble ICN system.” In: *Proceedings of conference on Organizational computing systems*. Milpitas, California, United States: ACM, pp. 130–137. DOI: [10.1145/224019.224032](https://doi.org/10.1145/224019.224032) (cit. on p. 12).
- Bogia, D. and Kaplan, S. (1995). “Flexibility and control for dynamic workflows in the WORLDS environment.” In: *Proceedings of conference on Organizational computing systems*. Milpitas, California, United States: ACM, pp. 148–159. DOI: [10.1145/224019.224034](https://doi.org/10.1145/224019.224034) (cit. on p. 13).

- Booth, W., Colomb, G., and Williams, J. (2008). *The Craft of Research*. 3rd ed. The University of Chicago Press. ISBN: 978-0-226-06566-3 (cit. on p. 9).
- Boyd, J. (1986). *Patterns of Conflict*. Tech. rep. Unpublished. URL: <http://danford.net/boyd/patterns.pdf> (cit. on p. 37).
- Buchholz, W. (1953). "The system design of the IBM Type 701 computer." In: *Proceedings of the IRE* 41.10, 1262–1275. DOI: [10.1109/JRPROC.1953.274300](https://doi.org/10.1109/JRPROC.1953.274300) (cit. on p. 65).
- Bussler, C. (1999). "Enterprise wide workflow management." In: *Concurrency, IEEE* 7.3, pp. 32–43. DOI: [10.1109/4434.788777](https://doi.org/10.1109/4434.788777) (cit. on p. 18).
- Callahan, S., Freire, J., Santos, E., Scheidegger, C., Silva, C., and Huy, T. (2006a). "Managing the Evolution of Dataflows with VisTrails." In: *Proceedings of the 22nd International Conference on Data Engineering*. DOI: [10.1109/ICDEW.2006.75](https://doi.org/10.1109/ICDEW.2006.75) (cit. on pp. 20, 22).
- (2006b). "VisTrails: visualization meets data management." In: *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, 745–747. DOI: [10.1145/1142473.1142574](https://doi.org/10.1145/1142473.1142574) (cit. on p. 22).
- Canonical (2012). *Ubuntu*. URL: <http://www.ubuntu.com/> (visited on 2012-08-06) (cit. on p. 128).
- Carroll, L. (2008). *Alice's Adventures in Wonderland (Project Gutenberg Edition)*. Project Gutenberg. URL: <http://www.gutenberg.org/ebooks/11> (cit. on p. 3).
- Casati, F., Ceri, S., Pernici, B., and Pozzi, B. (1998). "Workflow evolution." In: *Data & Knowledge Engineering* 24.3, pp. 211–238. DOI: [10.1016/S0169-023X\(97\)00033-5](https://doi.org/10.1016/S0169-023X(97)00033-5) (cit. on p. 13).
- Chalmers, A. (1999). *What Is This Thing Called Science?* 3rd ed. University of Queensland Press. ISBN: 0702230936 (cit. on p. 36).
- Chemboli, S. (2010a). *Contextualizing learning outcomes and course design in Moodle*. Presented at Moodleposium AU 2010. Canberra, Australia. URL: <http://hdl.handle.net/1885/9279> (cit. on pp. xiii, 99).
- (2010b). *Omnispective Analysis and Reasoning: An epistemic approach to scientific workflows*. Presented at the CECS Seminar Series, Australian National University. Canberra, Australia. URL: <http://cecs.anu.edu.au/seminars/more/SID/2503> (cit. on pp. xiii, 26).

- (2010c). *Workflow for creating a chat activity in Moodle*. URL: <http://csrins.wordpress.com/2010/07/19/workflow-for-creating-a-chat-activity-in-moodle/> (visited on 2011-06-25) (cit. on p. 102).
 - (2012a). *Soma - Origami Folding Demo*. URL: <http://ubuntuone.com/7Zz2B3fWLh8azFFgpT9rxR> (cit. on p. 137).
 - (2012b). *Soma: An OAR Tool Prototype*. URL: <https://launchpad.net/soma-app> (visited on 2012-08-31) (cit. on p. 122).
- Chemboli, S. and Boughton, C. (2011). “Contextual Course Design with Omnispersive Analysis and Reasoning.” In: *Changing Demands, Changing Directions. Proceedings ascilite*. Ed. by Williams, G., Brown, N., Pittard, M., and Cleland, B. Hobart, pp. 210–219. URL: <http://www.leishman-associates.com.au/ascilite2011/downloads/papers/Chemboli-full.pdf> (cit. on pp. xiii, 80).
- Chemboli, S. and Boughton, C. (2012a). “Managing Large and Complex Systems with Omnispersive Analysis and Reasoning.” In: *Proceedings of SETE APCOSE 2012*. Brisbane, Australia. URL: <http://hdl.handle.net/1885/9009> (cit. on pp. xiii, 80).
- (2012b). “Omnispersive Analysis and Reasoning: A Framework for Managing Intellectual Concerns in Scientific Workflows.” In: *Proceedings of the 5th India Software Engineering Conference*. Kanpur, India, pp. 143–146. DOI: [10.1145/2134254.2134279](https://doi.org/10.1145/2134254.2134279) (cit. on pp. xiii, 80).
- Chemboli, S., Kane, L., and Johns-Boast, L. (2010). *Translating Learning Outcomes in Moodle*. Presented at Moodlemoot AU 2010. Melbourne, Victoria, Australia. URL: <http://ubuntuone.com/4RBjlozHyEyITuy21aDCfe> (cit. on p. xiii).
- Chin, G., Schuchardt, K., Myers, J., and Gracio, D. (2000). “Participatory Workflow Analysis: Unveiling Scientific Research Processes with Scientists.” In: *PDC 2000 Proceedings of the Participatory Design Conference*. Ed. by Cherkasky, T., Greenbaum, J., Mambrey, P., and Pors, J. New York, NY, USA: CPSR, pp. 30–39. URL: <http://collaboratory.emsl.pnl.gov/resources/publications/papers/workflow%20analysis.html> (cit. on p. 18).
- Chin Jr., G., Sivaramakrishnan, C., Critchlow, T., Schuchardt, K., and Ngu, A. (2011). “Scientist-Centered Workflow Abstractions via Generic Actors, Workflow Templates, and Context-Awareness for Groundwater Modeling and Analysis.” In: *2011 IEEE World Congress on Services (SERVICES)*. IEEE, pp. 176–183. DOI: [10.1109/SERVICES.2011.31](https://doi.org/10.1109/SERVICES.2011.31) (cit. on pp. 26, 27, 51).

- Churches, A. (2008). *Bloom's Digital Taxonomy*. URL: <http://edorigami.wikispaces.com/Bloom%27s+Digital+Taxonomy> (visited on 2011-08-11) (cit. on p. 100).
- Churches, D., Gombas, G., Harrison, A., Maassen, J., Robinson, C., Shields, M., Taylor, I., and Wang, I. (2006). "Programming scientific and distributed workflow with Triana services." In: *Concurrency and Computation: Practice and Experience* 18.10, pp. 1021–1037. DOI: [10.1002/cpe.992](https://doi.org/10.1002/cpe.992) (cit. on p. 20).
- CIO Council (2001). *A Practical Guide to Federal Enterprise Architecture v1.0*. Tech. rep. P00201. U.S. Government Accountability Office. URL: <http://www.gao.gov/products/P00201> (cit. on p. 152).
- Csapo, B. (2009). "The Scientific Foundations of Teaching and Learning." In: *Green Book: For the renewal of public education in Hungary*. Ed. by Fajekas, K., Kollo, J., and Varga, J. Trans. by Babarczy, A. Budapest: Ecostat Government Institute for Strategic Research of Economy and Society, pp. 227–244. ISBN: 978-963-06-6690-9 (cit. on p. 92).
- Curcin, V. and Ghanem, M. (2008). "Scientific workflow systems—can one size fit all?" In: *Proceedings of International Biomedical Engineering Conference, CIBEC 2008*. Cairo, pp. 1–9. DOI: [10.1109/CIBEC.2008.4786077](https://doi.org/10.1109/CIBEC.2008.4786077) (cit. on p. 21).
- Dawkins, R. (2012). *Jaipur Literary Festival Q&A Session*. URL: <http://youtu.be/O9E5By0P3Go> (cit. on p. v).
- De Roure, D., Goble, C., and Stevens, R. (2009). "The Design and Realisation of the my Experiment Virtual Research Environment for Social Sharing of Workflows." In: *Future Generation Computer Systems* 25.5, pp. 561–567. DOI: [10.1016/j.future.2008.06.010](https://doi.org/10.1016/j.future.2008.06.010) (cit. on p. 49).
- Deelman, E. and Gil, Y. (2006a). "Managing Large-Scale Scientific Workflows in Distributed Environments: Experiences and Challenges." In: *Second IEEE International Conference on e-Science and Grid Computing (e-Science'06)*. DOI: [10.1109/E-SCIENCE.2006.261077](https://doi.org/10.1109/E-SCIENCE.2006.261077) (cit. on p. 29).
- (2006b). *NSF Workshop on the Challenges of Scientific Workflows*. Tech. rep. Arlington, VA: National Science Foundation. URL: <http://www.isi.edu/nsf-workflows06> (cit. on p. 156).
- Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., Su, M., Vahi, K., and Livny, M. (2004). "Pegasus: Mapping Scientific Workflows onto the Grid." In: *European Across Grids Conference*, 11–20. DOI: [10.1007/978-3-540-28642-4_2](https://doi.org/10.1007/978-3-540-28642-4_2) (cit. on p. 21).

- Deelman, E., Gannon, D., Shields, M., and Taylor, I. (2009). "Workflows and e-Science: An overview of workflow system features and capabilities." In: *Future Generation Computer Systems* 25.5, pp. 528–540. DOI: [10.1016/j.future.2008.06.012](https://doi.org/10.1016/j.future.2008.06.012) (cit. on p. 29).
- Demaine, E. and O'Rourke, J. (2007). *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press. ISBN: 9780521857574 (cit. on p. 82).
- Dijkstra, E. (1982). "On the role of scientific thought." In: *Selected Writings on Computing: A Personal Perspective*. New York: Springer-Verlag, pp. 60–66. ISBN: 0387906525 (cit. on p. 59).
- DoD CIO (2010). *DoD Architecture Framework Version 2.02*. URL: <http://dodcio.defense.gov/dodaf20.aspx> (cit. on pp. 148, 151).
- DoD CIO, Architecture & Infrastructural Directorate (2012). *DoDAF Version 2.0 Plenary*. Tech. rep. Mclean, Virginia. URL: http://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF_2-0_Plenary_Minutes_2012-01-05_V2.docx (cit. on p. 151).
- Drechsler, M. (2011). *Moodle Structural Overview*. URL: <http://www.slideshare.net/mark.drechsler/moodle-structural-overview> (cit. on p. 95).
- Ellis, C., Keddara, K., and Rozenberg, G. (1995). "Dynamic change within workflow systems." In: *Proceedings of conference on Organizational computing systems*. Milpitas, California, United States: ACM, pp. 10–21. DOI: [10.1145/224019.224021](https://doi.org/10.1145/224019.224021) (cit. on p. 13).
- Feynman, R. (1982). "Simulating physics with computers." In: *International journal of theoretical physics* 21.6, pp. 467–488. ISSN: 0020-7748 (cit. on p. 29).
- Flint, S. (2006). "Aspect-Oriented Thinking – an approach to bridging the interdisciplinary divides." PhD thesis. The Australian National University. URL: <http://hdl.handle.net/1885/49328> (cit. on pp. 5, 59, 80, 99, 147).
- (2008). "Rethinking Systems Thinking." In: *Proceedings of the 14th ANZSYS Australia New Zealand Systems Society Conference*. Ed. by Cook, D. Perth, Australia. URL: <http://www.anzsys.org/anzsys08/papers/Shayne%20Flint%20Rethinking%20Systems%20Thinking.pdf> (cit. on p. 37).
- (2009). "A Conceptual Model of Software Engineering Research Approaches." In: *2009 Australian Software Engineering Conference*. Gold Coast, Australia, pp. 229–236. DOI: [10.1109/ASWEC.2009.42](https://doi.org/10.1109/ASWEC.2009.42) (cit. on p. 4).

- Frické, M. (2009). "The knowledge pyramid: a critique of the DIKW hierarchy." In: *Journal of Information Science* 35.2, pp. 131–142. DOI: [10.1177/0165551508094050](https://doi.org/10.1177/0165551508094050) (cit. on p. 41).
- Garces, R., Jesus, T. de, Cardoso, J., and Valente, P. (2009). "Open Source Workflow Management Systems: A Concise Survey." In: *2009 BPM and Workflow Handbook: Methods, Concepts, Case Studies and Standards in Business Process Management and Workflow*, pp. 333–346. ISBN: 978-0-9777527-9-9 (cit. on p. 10).
- Garlan, D. and Shaw, M. (1993). "An Introduction to Software Architecture." In: *Advances in Software Engineering and Knowledge Engineering*. Ed. by Ambriola, V. and Tortora, G. Vol. 2. New Jersey: World Scientific Publishing Company, pp. 1–40. ISBN: 978-9810215941 (cit. on p. 148).
- Gray, H. (1918). *Anatomy of the Human Body*. Lea & Febiger, 1918; Bartleby.com, 2000. URL: <http://www.bartelby.com/107/> (cit. on p. 59).
- Han, J., Kamber, M., and Gray, J. (2000). *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers Inc. ISBN: 1-55860-489-8 (cit. on p. 14).
- Han, Y., Sheth, A., and Bussler, C. (1998). "A Taxonomy of Adaptive Workflow Management." In: *1998 ACM Conference on Computer Supported Cooperative Work (CSCW-98)*. Seattle, WA, USA (cit. on p. 18).
- Hartmanis, J. (1995). "Turing Award Lecture: On Computational Complexity and the Nature of Computer Science." In: *ACM Computing Surveys* 27.1. DOI: [10.1145/194313.214781](https://doi.org/10.1145/194313.214781) (cit. on p. 79).
- Hollingsworth, D. (1995). *The workflow reference model*. Tech. rep. Document Number TCo0-1003. UK: The Workflow Management Coalition. URL: <http://www.wfmc.org/standards/docs/tc003v11.pdf> (cit. on pp. 10, 15, 16).
- Houghton, W. (2004). *Engineering subject centre guide: Learning and teaching theory for engineering academics*. Loughborough: HEA Engineering Subject Centre. URL: <http://www.engsc.ac.uk/learning-and-teaching-theory-guide> (cit. on p. 95).
- Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M., Li, P., and Oinn, T. (2006). "Taverna: a tool for building and running workflows of services." In: *Nucleic Acids Research* 34, W729–W732. DOI: [10.1093/nar/gkl320](https://doi.org/10.1093/nar/gkl320) (cit. on p. 22).
- IEEE (2012). *Survey of Architecture Frameworks*. URL: <http://www.iso-architecture.org/ieee-1471/afs/frameworks-table.html> (visited on 2012-06-29) (cit. on p. 151).
- Ioannidis, Y., Livny, M., Ailamaki, A., Narayanan, A., and Therber, A. (1997). "Zoo: a desktop experiment management environment." In: *Proceedings of the 1997*

- ACM SIGMOD international conference on Management of data*. Tucson, Arizona, United States: ACM, pp. 580–583. DOI: [10.1145/253260.253415](https://doi.org/10.1145/253260.253415) (cit. on pp. 11, 15).
- Iordan, V. and Cicortas, A. (2008). “Considerations on Using Ontologies in Complex Systems.” In: *10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, 2008. SYNASC '08*. IEEE, pp. 305–309. DOI: [10.1109/SYNASC.2008.78](https://doi.org/10.1109/SYNASC.2008.78) (cit. on p. 107).
- ISO/IEC/IEEE (2011). *ISO/IEC/IEEE 42010 - Systems and software engineering – Architecture description*. Standard. Institute of Electrical and Electronics Engineers Inc. URL: <http://www.iso-architecture.org/ieee-1471/> (cit. on p. 149).
- Jamshidi, M. (2008). “System of systems engineering - New challenges for the 21st century.” In: *IEEE Aerospace and Electronic Systems Magazine* 23.5, pp. 4–19. DOI: [10.1109/MAES.2008.4523909](https://doi.org/10.1109/MAES.2008.4523909) (cit. on p. 104).
- Kenneway, E. (1987). *Complete Origami*. London: Ebury Press. ISBN: 0852236174 (cit. on p. 82).
- Lin, C., Lu, S., Fei, X., Chebotko, A., Pai, D., Lai, Z., Hua, J., and Foutouhi, F. (2009). “A Reference Architecture for Scientific Workflow Management Systems and the View SOA Solution.” In: *IEEE Transactions on Services Computing* 2.1, pp. 79–92. DOI: [10.1109/TSC.2009.4](https://doi.org/10.1109/TSC.2009.4) (cit. on p. 17).
- Ludäscher, B., Bowers, S., and McPhillips, T. (2009). “Scientific Workflows.” In: *Encyclopedia of Database Systems*, pp. 2507–2511. URL: http://dx.doi.org/10.1007/978-0-387-39940-9_1471 (cit. on p. 20).
- Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E., Tao, J., and Zhao, Y. (2006). “Scientific workflow management and the Kepler system.” In: *Concurrency and Computation: Practice and Experience* 18.10, pp. 1039–1065. DOI: [10.1002/cpe.994](https://doi.org/10.1002/cpe.994) (cit. on pp. 21, 24).
- Ludäscher, B., Weske, M., McPhillips, T., and Bowers, S. (2009). “Scientific workflows: Business as usual?” In: *Business Process Management*, 31–47. DOI: [10.1007/978-3-642-03848-8_4](https://doi.org/10.1007/978-3-642-03848-8_4) (cit. on pp. 10, 24, 45, 54).
- Marcus, A. (2011). *PNAS retracts two papers on osmolytes after researchers discover crucial measurement errors*. URL: <http://retractionwatch.wordpress.com/2011/12/15/pnas-retracts-two-papers-on-osmolytes-after-researchers-discover-crucial-measurement-errors/> (visited on 2011-12-18) (cit. on p. 26).
- Marsh, C. (2009). *Key Concepts for Understanding Curriculum*. 4th ed. Routledge. ISBN: 978-0415465786 (cit. on p. 94).

- Maus, H. (2001). "Workflow Context as a Means for Intelligent Information Support." In: *Modeling and Using Context* 2116. Ed. by Akman, V., Bouquet, P., Thomason, R., and Young, R., pp. 261–274. DOI: [10.1007/3-540-44607-9_20](https://doi.org/10.1007/3-540-44607-9_20) (cit. on p. 26).
- McClatchey, R., Estrella, F., Goff, J. le, Kovacs, Z., and Baker, N. (1997). "Object Databases in a Distributed Scientific Workflow Application." In: *Proceedings of the 3rd Basque International Workshop on Information Technology (BIWIT'97)*. Biarritz, pp. 11–21. DOI: [10.1109/BIWIT.1997.614047](https://doi.org/10.1109/BIWIT.1997.614047) (cit. on p. 15).
- McPhillips, T., Bowers, S., Zinn, D., and Ludäscher, B. (2009). "Scientific workflow design for mere mortals." In: *Future Generation Computer Systems* 25.5, 541–551. DOI: [10.1016/j.future.2008.06.013](https://doi.org/10.1016/j.future.2008.06.013) (cit. on p. 21).
- Medeiros, C., Vossen, G., and Weske, M. (1995). "WASA: A workflow-based architecture to support scientific database applications." In: *Database and Expert Systems Applications*. Lecture Notes in Computer Science 978. Ed. by Revell, N. and Tjoa, A., pp. 574–583. DOI: [10.1007/BFb0049154](https://doi.org/10.1007/BFb0049154) (cit. on pp. 13, 14).
- Medina-Mora, R., Winograd, T., Flores, R., and Flores, F. (1992). "The action workflow approach to workflow management technology." In: *Proceedings of the 4th Conference on Computer-supported Cooperative Work*. ACM New York, NY, USA, pp. 281–288. DOI: [10.1145/143457.143530](https://doi.org/10.1145/143457.143530) (cit. on pp. 10, 11).
- Meidanis, J., Vossen, G., and Weske, M. (1996). "Using Workflow Management in DNA Sequencing." In: *Proceedings of the First IFCIS International Conference on Cooperative Information Systems*. Brussels: IEEE Computer Society, pp. 114–123. DOI: [10.1109/COOPIS.1996.555003](https://doi.org/10.1109/COOPIS.1996.555003) (cit. on pp. 14, 24).
- Mellor, S. and Balcer, M. (2002). *Executable UML: A foundation for model-driven architecture*. Addison-Wesley. ISBN: 978-0201748048 (cit. on p. 5).
- Mitchell, M. (2009). *Complexity: A Guided Tour*. Oxford University Press. ISBN: 9780195124415 (cit. on p. 106).
- Moodle Community (2012). *Moodle.org*. URL: <http://moodle.org/> (visited on 2012-02-03) (cit. on p. 95).
- National Academy of Sciences (2011). "Retraction for Dougan et al., Solvent molecules bridge the mechanical unfolding transition state of a protein." In: *Proceedings of the National Academy of Sciences*. DOI: [10.1073/pnas.1118431108](https://doi.org/10.1073/pnas.1118431108) (cit. on p. 26).
- National Research Council (2002). *Scientific Research in Education: Committee on Scientific Principles for Education Research*. Ed. by Shavelson, R. and Towne,

- L. Washington, D.C.: Center for Education. Division of Social Sciences and Education. National Academy Press. ISBN: 0-309-08291-9 (cit. on p. 92).
- Ngu, A., Jamnagarwala, A., Chin Jr., G., Sivaramakrishnan, C., and Critchlow, T. (2010). "Context-aware scientific workflow systems using KEPLER." In: *International Journal of Business Process Integration and Management* 5.1. Ed. by Lu, S., Deelman, E., and Zhao, Z., pp. 18–31. DOI: [10.1504/IJBPIIM.2010.033172](https://doi.org/10.1504/IJBPIIM.2010.033172) (cit. on pp. 27, 51).
- Ngu, A., Jamnagarwala, A., Chin, G., Sivaramakrishnan, C., and Critchlow, T. (2011). "Kepler Scientific Workflow Design and Execution with Contexts." In: *International Journal of Computers and Their Applications* 18.3. Ed. by Chebotko, A., Simmhan, Y., and Missier, P., pp. 133–147. ISSN: 1076-5204 (cit. on p. 51).
- Oinn, T., Addis, M., Ferris, J., Marvin, D., Greenwood, M., Goble, C., Wipat, A., Li, P., and Carver, T. (2004a). "Delivering web service coordination capability to users." In: *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. New York, NY, USA, 438–439. DOI: [10.1145/1013367.1013514](https://doi.org/10.1145/1013367.1013514) (cit. on p. 23).
- Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M., Wipat, A., and Li, P. (2004b). "Taverna: a tool for the composition and enactment of bioinformatics workflows." In: *Bioinformatics (Oxford, England)* 20.17, pp. 3045–3054. DOI: [10.1093/bioinformatics/bth361](https://doi.org/10.1093/bioinformatics/bth361) (cit. on pp. 20, 22).
- Oinn, T., Greenwood, M., Addis, M., Alpdemir, M., Ferris, J., Glover, K., Goble, C., Goderis, A., Hull, D., Marvin, D., Li, P., Lord, P., Pocock, M., Senger, M., Stevens, R., Wipat, A., and Wroe, C. (2006). "Taverna: lessons in creating a workflow environment for the life sciences." In: *Concurrency and Computation: Practice and Experience* 18.10, pp. 1067–1100. DOI: [10.1002/cpe.993](https://doi.org/10.1002/cpe.993) (cit. on p. 22).
- O'Neill, G. (2010). *Program Design: Overview of curriculum models*. Tech. rep. University College Dublin. URL: <http://www.ucd.ie/t4cms/UCDTLP00631.pdf> (cit. on p. 94).
- Op't Land, M., Proper, E., Waage, M., Jeroen, C., and Claudia, S. (2009). *Enterprise Architecture: Creating Value by Informed Governance*. Springer. ISBN: 978-3-540-85231-5 (cit. on p. 151).
- Parnas, D. (1972). "On the Criteria to be used in Decomposing Systems into Modules." In: *Communications of the ACM* 15.12. Ed. by Morris, R., pp. 1053–1058. DOI: [10.1145/361598.361623](https://doi.org/10.1145/361598.361623) (cit. on pp. 59, 121).

- Peirce, C. (1877). "The Fixation of Belief." In: *Popular Science Monthly* 12, pp. 1–15. URL: <http://www.peirce.org/writings/p107.html> (cit. on pp. 38, 39, 146).
- Pereira, W. and Travassos, G. (2010). "Towards the conception of scientific workflows for in silico experiments in software engineering." In: *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. DOI: 10.1145/1852786.1852831 (cit. on p. 27).
- Pham, A. and Pham, P. (2012). *Scrum in Action: Agile Software Project Management and Development*. Course Technology. ISBN: 978-1-4354-5913-7 (cit. on pp. 122, 127).
- Pignotti, E., Edwards, P., Gotts, N., and Polhill, G. (2011). "Enhancing workflow with a semantic description of scientific intent." In: *Web Semantics: Science, Services and Agents on the World Wide Web* 9.2. Ed. by Gil, Y. and Groth, P., pp. 222–244. DOI: 10.1016/j.websem.2011.05.001 (cit. on p. 26).
- Python Software Foundation (2012). *Python Programming Language*. URL: <http://www.python.org/> (visited on 2012-08-06) (cit. on p. 128).
- Ramsden, P. (2003). *Learning to Teach in Higher Education*. 2nd ed. Routledge. ISBN: 978-0415303453 (cit. on p. 97).
- Reuß, T., Vossen, G., and Weske, M. (1997). "Modeling samples processing in laboratory environments as scientific workflows." In: *Proceedings of the Eighth International Workshop on Database and Expert Systems Applications, 1997*, pp. 49–54. DOI: 10.1109/DEXA.1997.617233 (cit. on p. 15).
- Rowley, J. (2007). "The wisdom hierarchy: representations of the DIKW hierarchy." In: *Journal of Information Science* 33.2, pp. 163–180. DOI: 10.1177/0165551506070706 (cit. on p. 41).
- Sage, A. and Cuppan, C. (2001). "On the Systems Engineering and Management of Systems of Systems and Federations of Systems." In: *Information, Knowledge, Systems Management* 2.4, 325–345. ISSN: 1389-1995 (cit. on p. 104).
- Sahoo, S., Sheth, A., and Henson, C. (2008). "Semantic Provenance for eScience: Managing the Deluge of Scientific Data." In: *Internet Computing, IEEE* 12.4. Ed. by Blake, M. and Huhns, M., pp. 46–54. DOI: 10.1109/MIC.2008.86 (cit. on pp. 29, 107).
- Scharfstein, B. (1991). *The Dilemma of Context*. NYU Press. ISBN: 978-0814779163 (cit. on p. 27).

- Schill, A. and Mittasch, C. (1996). "Workflow management systems on top of OSF DCE and OMG CORBA." In: *Distributed Systems Engineering* 3.4. Ed. by Chrysanthos, P., pp. 250–262. DOI: [10.1088/0967-1846/3/4/005](https://doi.org/10.1088/0967-1846/3/4/005) (cit. on p. 14).
- Schwaber, K. and Sutherland, J. (2011). *The Scrum Guide*. Scrum.org. URL: http://www.scrum.org/storage/scrumguides/Scrum_Guide.pdf (cit. on p. 122).
- Sessions, R. (2006). *A Better Path to Enterprise Architectures*. Objectwatch Inc. URL: <http://ww.objectwatch.com/whitepapers/ABetterPath-Final.pdf> (cit. on pp. 59, 147).
- Sheard, S. and Mostashari, A. (2009). "Principles of complex systems for systems engineering." In: *Systems Engineering* 12.4, pp. 295–311. DOI: [10.1002/sys.20124](https://doi.org/10.1002/sys.20124) (cit. on p. 104).
- Shi, M., Yang, G., Xiang, Y., and Wu, S. (1998). "Workflow Management Systems: A Survey." In: *Proceedings of the International Conference on Communication Technology (ICCT '98)*. DOI: [10.1109/ICCT.1998.740974](https://doi.org/10.1109/ICCT.1998.740974) (cit. on p. 17).
- Shields, M. (2007). "Control- Versus Data-Driven Workflows." In: *Workflows for e-Science: Scientific Workflows for Grids*. Ed. by Taylor, I., Deelman, E., Gannon, D., and Shields, M. Springer London, pp. 167–173. ISBN: 978-1-84628-519-6 (cit. on p. 21).
- Shoshani, A., Olken, F., and Wong, H. (1984). "Characteristics of Scientific Databases." In: *Proceedings of the 10th International Conference on Very Large Data Bases*. Morgan Kaufmann Publishers Inc., pp. 147–160. ISBN: 0-934613-16-8 (cit. on pp. 10, 11).
- Shuell, T. (1986). "Cognitive Conceptions of Learning." In: *Review of Educational Research* 56.4, pp. 411–436. DOI: [10.2307/1170340](https://doi.org/10.2307/1170340) (cit. on p. 91).
- Shuttleworth, M. (2012). *Re: No more dodge windows in Unity?* URL: <https://lists.launchpad.net/unity-design/msg07680.html> (visited on 2012-02-09) (cit. on p. 103).
- Sickafus, E. (1997). *Unified structured inventive thinking: How to invent*. Ntelleck. ISBN: 096594350X (cit. on p. 147).
- Simon, H. (1956). "Rational choice and the structure of the environment." In: *Psychological review* 63.2, p. 129. ISSN: 1939-1471 (cit. on p. 70).
- Singh, M. and Vouk, M. (1996). "Scientific Workflows: Scientific Computing Meets Transactional Workflows." In: *Proceedings of the NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-Art and Future Directions*,

- pp. 28–34. URL: <http://www.csc.ncsu.edu/faculty/mpsingh/papers/databases/workflows/sciworkflows.html> (cit. on pp. 10, 18, 19, 45, 54).
- Sneider, R. and Larner, K. (2009). *The Art of Being a Scientist: A Guide for Graduate Students and Their Mentors*. Cambridge University Press. ISBN: 9780521743525 (cit. on pp. 37, 48).
- SQLite (2012). *SQLite Database Engine*. URL: <http://www.sqlite.org/> (visited on 2012-07-30) (cit. on p. 128).
- Staab, S. and Studer, R., eds. (2009). *Handbook on Ontologies*. 2nd ed. Springer. ISBN: 3540709991 (cit. on p. 107).
- Steffen, W. (2008). *Surviving the Anthropocene: Changing Our Interaction with the Fragile Planet : Sensible Ways Forward in Response to Global Change*. Presented at the Fenner School Seminar Series, Australian National University. The Australian National University. URL: http://fennerschool.anu.edu.au/news_events/seminars/seminars_2008.php#16oct (cit. on p. 29).
- Sterman, J. (2002). “All models are wrong: reflections on becoming a systems scientist.” In: *System Dynamics Review* 18.4, pp. 501–531. DOI: 10.1002/sdr.261 (cit. on p. 71).
- Strazdins, P. (2007). *Research-Based Education in Computer Science at the ANU: Challenges and Opportunities*. Tech. rep. The Australian National University. URL: <http://cs.anu.edu.au/techreports/2007/TR-CS-07-05.pdf> (cit. on p. 97).
- Tang, A., Han, J., and Chen, P. (2004). “A comparative analysis of architecture frameworks.” In: *Proceedings of the 11th Asia-Pacific Software Engineering Conference, 2004*, 640–647. DOI: 10.1109/APSEC.2004.2 (cit. on p. 150).
- Tharp, T. and Reiter, M. (2006). *The Creative Habit*. Simon and Schuster. ISBN: 9780743235273 (cit. on p. 81).
- The Adventures of Sherlock Holmes: The Red-Headed League* (1985). Granada Television (cit. on p. 141).
- The GNOME Project (2012). *PyGObject - GLib/GObject/GIO Python bindings*. URL: <https://live.gnome.org/PyGObject> (visited on 2012-08-06) (cit. on p. 128).
- Tichy, W. (1998). “Should Computer Scientists Experiments More?” In: *IEEE Computer* 31.5, pp. 32–40. DOI: 10.1109/2.675631 (cit. on p. 79).
- Truong, H., Fahringer, T., Nerieri, F., and Dustdar, S. (2005). “Performance metrics and ontology for describing performance data of grid workflows.” In: *IEEE*

- International Symposium on Cluster Computing and the Grid, 2005. CCGRID 2005*. Vol. 1. IEEE, pp. 301–308. DOI: [10.1109/CCGRID.2005.1558569](https://doi.org/10.1109/CCGRID.2005.1558569) (cit. on p. 107).
- Turuncoglu, U. (2012). “Applying Scientific Workflow to ESM.” In: *Earth System Modelling - Volume 5 - Tools for Configuring, Building and Running Models*. Ed. by Ford, R., Riley, G., Budich, R., and Redler, R. Vol. 5. SpringerBriefs in Earth System Sciences. Springer Heidelberg, pp. 15–29. ISBN: 978-3-642-23931-1 (cit. on p. 26).
- van der Aalst, W. and Basten, T. (2002). “Inheritance of workflows: an approach to tackling problems related to change.” In: *Theoretical Computer Science* 270.1-2, pp. 125–203. DOI: [doi:10.1016/S0304-3975\(00\)00321-2](https://doi.org/doi:10.1016/S0304-3975(00)00321-2) (cit. on p. 13).
- van der Aalst, W., Barthelmess, P., Ellis, C., and Wainer, J. (2000). “Workflow modeling using procllets.” In: *Cooperative Information Systems*. Ed. by Scheuermann, P. and Etzion, O. Vol. 1901. Lecture Notes in Computer Science, 198–209. DOI: [10.1007/10722620_20](https://doi.org/10.1007/10722620_20) (cit. on p. 18).
- van der Aalst, W., Barthelmess, P., Eliis, C., and Wainer, J. (2001). “Procllets: A framework for lightweight interacting workflow processes.” In: *International Journal of Cooperative Information Systems* 10.4, 443–482. ISSN: 0218-8430 (cit. on p. 18).
- Wainer, J., Weske, M., Vossen, G., and Medeiros, C. (1997). “Scientific Workflow Systems.” In: *Proceedings of the NSF Workshop on Workflow and Process Automation in Information Systems: State of the Art and Future Directions* (cit. on pp. 10, 19, 20).
- Walker, D. (1971). “A Naturalistic Model for Curriculum Development.” In: *The School Review* 80.1, pp. 51–65. ISSN: 0036-6773. URL: <http://www.jstor.org/stable/1084221> (cit. on p. 94).
- Wang-Iverson, P., Lang, R., and Yim, M., eds. (2011). *Origami 5*. 1st ed. A K Peters/CRC Press. ISBN: 9781568817149 (cit. on p. 82).
- Warfield, J. (1994). *Science of Generic Design: Managing Complexity Through Systems Design*. 2nd ed. Iowa State University Press. ISBN: 0813822475 (cit. on pp. 35, 39–43, 46–48, 53, 57, 71, 105, 106).
- Weske, M., Vossen, G., and Medeiros, C. (1996). *Scientific Workflow Management: WASA Architecture and Applications*. Tech. rep. (cit. on pp. 14, 21).
- Wiener, J. and Ioannidis, Y. (1993). “A moose and a fox can aid scientists with data management problems.” In: *Proceedings of the International Workshop on Database Programming Languages*, 376–398 (cit. on p. 12).

COLOPHON

This document was typeset in \LaTeX using the typographical look-and-feel *arsclassica* developed by Lorenzo Pantieri, which is a customization of the *Classic-Thesis* style designed by André Miede. The text font is Palatino; the heading font is Iwona; and the font in figures is Ubuntu Regular.

Final Render as of September 26, 2013 (version 12.09.04).