

Problem Map: A Framework for Investigating the Role of Problem Formulation
in Creative Design

by

Mahmoud Dinar

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved July 2015 by the
Graduate Supervisory Committee:

Jami Shah, Chair
Pat Langley
Joseph Davidson
Micah Lande
Yi Ren

ARIZONA STATE UNIVERSITY

August 2015

ABSTRACT

Design problem formulation is believed to influence creativity, yet it has received only modest attention in the research community. Past studies of problem formulation are scarce and often have small sample sizes. The main objective of this research is to understand how problem formulation affects creative outcome. Three research areas are investigated: development of a model which facilitates capturing the differences among designers' problem formulation; representation and implication of those differences; the relation between problem formulation and creativity.

This dissertation proposes the Problem Map (P-maps) ontological framework. P-maps represent designers' problem formulation in terms of six groups of entities (requirement, use scenario, function, artifact, behavior, and issue). Entities have hierarchies within each group and links among groups. Variables extracted from P-maps characterize problem formulation.

Three experiments were conducted. The first experiment was to study the similarities and differences between novice and expert designers. Results show that experts use more abstraction than novices do and novices are more likely to add entities in a specific order. Experts also discover more issues.

The second experiment was to see how problem formulation relates to creativity. Ideation metrics were used to characterize creative outcome. Results include but are not limited to a positive correlation between adding more issues in an unorganized way with quantity and variety, more use scenarios and functions with novelty, more behaviors and conflicts identified with quality, and depth-first exploration with all ideation metrics.

Fewer hierarchies in use scenarios lower novelty and fewer links to requirements and issues lower quality of ideas.

The third experiment was to see if problem formulation can predict creative outcome. Models based on one problem were used to predict the creativity of another. Predicted scores were compared to assessments of independent judges. Quality and novelty are predicted more accurately than variety, and quantity. Backward elimination improves model fit, though reduces prediction accuracy.

P-maps provide a theoretical framework for formalizing, tracing, and quantifying conceptual design strategies. Other potential applications are developing a test of problem formulation skill, tracking students' learning of formulation skills in a course, and reproducing other researchers' observations about designer thinking.

DEDICATION

To my parents and my sister Sana.

ACKNOWLEDGMENTS

This dissertation was written in the department of Mechanical and Aerospace Engineering at Arizona State University in Tempe Arizona throughout the spring of 2015. It was written under the supervision of my esteemed adviser Professor Jami Shah. I wish to express my deepest gratitude for all of his help, advice, and encouragement during the years I studied and did research in the Design Automation Lab. Along the way, he involved me in discussions around a variety of other projects, publications, and proposals which taught me many great lessons about competence, merit, ethics, and ambition in an academic life. His direct, yet humble and honest approach in communicating with students is a character which I wish to emulate in my future career, if I can. It has been a great honor for me to have the privilege to work with him and I will forever be indebted to him.

I cannot thank enough my adviser and friend, Professor Pat Langley. His meticulous knowledge of a variety of subjects and his sharp attention to details have surprised me in many occasions about how easily I might make mistakes if I express an idea carelessly. His passionate arguments during many of our conversations have kept me enthusiastic about what I do. His curiosity has taught me that life is about learning. He will forever be an inspiration.

I should thank my other committee members, Professor Joseph Davidson, Dr. Micah Lande, and Dr. Yi Ren for their support and advice in preparing this dissertation. I should extend my gratitude to Professor George Runger, Dr. Kenneth Huebner, and Dr. Winslow

Burleson who were on my committee when I first proposed my thesis prospectus. I also thank Dr. Ellen Campana who was a part of our project in its earlier days.

My research was supported by the National Science Foundation, grant number 1002910. I thank the NSF for providing this opportunity for me to be part of a group of scientists who contributed to my understanding of some of the fundamental blocks of conceptual design thinking.

I should also thank all of my friends here in Arizona State University for their help, support, and friendship throughout these years, especially Chris Maclellan, Andreea Danielescu, Glen Hunt, Prashant Mohan, Shahrouz Sharifi, Benyamin Gholami, Nathan Kalish, Garen Minassians, and all my colleagues at the Design Automation Lab.

Finally, I should thank my family for being there for me and supporting me every step of the way throughout my life. My elderly parents never stopped encouraging me to cease the opportunities I had even though it meant being apart from them for many years. There is a person that has a special place in my heart, my wise and wonderful sister Sana. Without her, I would have never become the man I am today. Thank you my dear.

Mahmoud Dinar

Tempe, Arizona

July 2015

TABLE OF CONTENTS

	Page
LIST OF TABLES	x
LIST OF FIGURES	xii
CHAPTER	
1 INTRODUCTION	1
1.1 Motivation	1
1.2 Research Questions and Hypotheses	3
1.3 Research Tasks	5
1.4 Guide to the Dissertation	6
2 LITERATURE REVIEW	9
2.1 The Process of Problem Formulation in Design	11
2.1.1 Processes, Methods and Best Practices	11
2.1.2 Strategies	14
2.1.3 Differences between Novices and Experts	16
2.2 Representation frameworks and formalisms	16
2.2.1 Design Representations	17
2.2.2 Ontologies	25
2.2.3 Computer Formalisms	27
3 TOWARDS A STRUCTURED REPRESENTATION	31
3.1 Initial Modeling Structure	32
3.2 Modified Modeling Structure	40
3.3 Synthesizing the Models from the Exploratory Studies	47

CHAPTER	Page
3.4 Specifications of the Modeling Framework	50
4 THE PROBLEM MAP ONTOLOGICAL FRAMEWORK	55
4.1 Initial Data Model	55
4.2 Improved Data Model.....	57
4.3 Model Validation	63
5 THE PROBLEM FORMULATOR TESTBED	70
5.1 System Architecture.....	70
5.2 Graphical User Interface	73
5.3 Test and User Studies.....	75
5.4 Improvements and Added Features	78
6 EMPIRICAL STUDIES - PRELIMINARIES	86
6.1 Characteristics of Design Problems.....	87
6.2 Characteristics of Participating Designers	91
6.3 Characteristics of Problem Formulation	94
6.3.1 State Characteristics	94
6.3.2 Examples of state Characteristics	97
6.3.3 Process Characteristics (Strategies)	100
6.3.4 Examples of Strategies	103
6.4 Characteristics of Creative Outcome.....	106
6.5 Design of Experiments.....	110
7 EXPERIMENT I: DIFFERENCES IN EXPERTS AND NOVICES	114
7.1 Collected Data.....	115

CHAPTER	Page
7.2 Analysis Method	117
7.3 Results and Conclusions	119
7.3.1 Representing Differences within Experts	119
7.3.2 Representing Differences within Novices	122
7.3.3 Testing Differences between Experts and Novices	126
8 EXPERIMENT II: RELATING FORMULATION TO CREATIVITY	130
8.1 Collected Data.....	131
8.2 Analysis Method	132
8.3 Results and Conclusions	137
8.3.1 Correlation Analysis.....	137
8.3.2 Regression Analysis	141
8.3.3 Improving Model Fit with Backward Elimination	144
8.3.4 Classification with Decision Trees	146
8.3.5 Examining Progress in Creativity	154
9 EXPERIMENT III: PREDICTING CREATIVITY FROM FORMULATION ..	159
9.1 Collected Data.....	159
9.2 Analysis Method	160
9.3 Results and Conclusions	161
10 POTENTIAL APPLICATIONS	170
10.1 Applied Test of Problem Formulation Skill.....	170
10.1.1 Identification of Subskills	171
10.1.2 Associating P-maps Measures with Sub-skills.....	174

CHAPTER	Page
10.1.3 Candidate Test Items	176
10.2 Objective Evaluation of Students' Problem Formulation.....	177
10.3 A Vehicle for Reproducing Previous Studies	182
11 CONCLUDING REMARKS	184
11.1 Research Questions Revisited	184
11.2 Limitations	189
11.2.1 Limitations of the Exploratory Studies	189
11.2.2 Limitations of the Experimental Studies	192
11.3 Future Work	194
11.4 Original Contributions	196
11.5 Publications	198
REFERENCES.....	201
APPENDIX	
A DESIGNERS' FORMULATION SHOWN IN SNAPSHOTS	210
B ASP ENCODINGS OF STRATEGIES	214
C EXCERTPS OF A CODED PROTOCOL (DP_1)	225
D REGRESSORS OF STATE COUNTS MODELS	233
E REGRESSORS AFTER BACKWARD ELIMINATION	236
F HISTOGRAMS OF PREDICTION RESIDUALS	240

LIST OF TABLES

Table	Page
3.1 Coded Segments from the First Protocol Analysis.....	35
3.2 Coded Segments from the First Protocol Analysis (from [81]).....	44
3.3 Specifications of a Framework for Problem Formulation.....	52
3.4 Measures of Goodness for the Tentative Framework	54
4.1. Comparison of Different Modeling Frameworks to P-Maps	64
4.2 A Protocol Coded in F-B-S [35] Compared to P-Maps (from [2]).....	66
5.1 Results of a User Study on the First Problem formulator	77
6.1 Summary of the Design of Experiments.....	87
6.2 Distribution of Participants' Divergent Thinking Test [6] Scores	93
6.3 Examples of P-Maps State Characteristics	95
6.4 An Example of State Counts for a P-Map	100
6.5 List of formalized Problem Formulation Strategies	102
6.6 A Sample Concept inventory for the Gopher Problem (DP_3)	110
6.7 The Design of Experiments.....	113
7.1 Change in Novices' Time of Discovering Issues Through Practice	124
7.2 Change in the % of Issues Novices Discover Through Practice.....	124
7.3 Frequent Sub-Sequences with a Support Higher Than 50%	126
7.4 Variations in Adopting Two Strategies Among Students and Experts.....	127
7.5 Differences Between Experts and Novices in the Amount of Issues.....	127
7.6 Differences Between Experts and Novices in the Time of adding Issues	128
8.1 Correlations Between DT Test and P-Maps for Experts (from [108])	139

Table	Page
8.2 Significant Formulation-Ideation Correlations for Students.....	140
8.3 Regressors of P-Maps Strategies Counts Models for Two Problems.....	142
8.4 Test of Model Fit with R^2	143
8.5 Improvements in Model Fit after Backward Elimination	146
8.6 Comparison of Decision Trees Built for Quantity	149
8.7 Comparison of Decision Trees Built for Variety	151
8.8 Comparison of Decision Trees Built for Novelty.....	152
8.9 Comparison of Decision Trees Built for Quality	153
8.10 Changes in Ideation Metrics for a Class as a Whole.....	155
8.11 Changes in individuals' Ideation Metrics for a Class.....	156
9.1 Accuracy of Predicting Dp_5 Ideation with Dp_4 Regression Models	162
9.2 Accuracy of Predicting Ideation after Backward Elimination	163
9.3 Differences of actual and Predicted Ideation; Mean Row 1; <i>P</i> Value Row 2	168
10.1 P-Maps Measures for PF Subskills.....	172
10.2 Examples of Implicit And Fictitious Requirements inventory	175
10.3 Examples of Key and Irrelevant Issues inventory.....	177
10.4 A Rubric for Evaluating Students' PF in a Design Task	178
10.5 The Scoring Scheme for Evaluating Students' PF in a Design Task.....	179
10.6 Test of Changes in Individuals' Problem Formulation Sub-Skills	181

LIST OF FIGURES

Figure	Page
2.1 A Simplistic Model of Design (Adapted from [14])	10
2.2 A Model of Activities in Design in the F-B-S Framework (from [48])	19
2.3 Function-Behavior-State Diagram (from [55])	21
2.4 The Four-Box Diagram (from [59])	22
2.5 An Example of Evaluating A BID Analogue in The Four-Box (from [59])....	23
2.6 A Concept Map of Formulating a Design Problem	29
3.1 The Design Task of The First Exploratory Protocol Study	33
3.2 The Problem Formulation Ontology from the First Exploratory Study.....	37
3.3 A Snapshot of the Novices Halfway Through Their Session (from [77])	39
3.4 A Snapshot of the Novices at the End of Their Session (from [77]).....	39
3.5 A Snapshot of the Expert at the End of His Design Session (from [77]).....	40
3.6 Expert's Formulation after 8 (A) And 13 (B) Minutes (from [81])	45
3.7 Novice's Formulation after 11 (A) And 17 (B) Minutes (from [81])	46
3.8 A Collection of Entities from Brainstorming.....	48
3.9 Merging Entities in Multiple Steps.....	50
4.1 The First Structured P-Maps Framework.....	56
4.2 The Data Model for the Updated P-Maps Ontology.....	58
5.1 The System Architecture of the Problem Formulator (from [95])	72
5.2 Database Schema for the Problem Formulator (from [95]).....	73
5.3 The Main GUI of the Problem Formulator	75

Figure	Page
5.4 The First GUI for Problem Formulator	76
5.5 Problem Formulator Enhancements - Tree View	79
5.6 Problem Formulator Enhancements – Collapsing Nodes	80
5.7 Problem Formulator Enhancements – Network View	80
5.8 Problem Formulator Enhancements – Objective Tree input	81
5.9 Problem Formulator Enhancements – Objective Tree Output	82
5.10 The Depth Exploration Approach	83
5.11 The Breadth Exploration Approach	84
5.12 Problem Formulator Enhancements – Retrospective Module.....	85
6.1 Problem Statement for Water Sampler (DP_1).....	88
6.2 Problem Statement for Can Crusher (DP_2).....	89
6.3. The Settings for the Goofy Gopher Problem (DP_3)	89
6.4. The Settings for the Shot Buddy Problem (DP_4).....	90
6.5. The Settings for the Autonomous Surveillance Problem (DP_5)	91
6.6 A Historic Sample of the Divergent Thinking Test Scores (from [99]).....	94
6.7 A Snapshot of a P-Map for the State Counts Example.....	98
6.8 Tree View for the State Counts Example.....	99
6.9 Network View of the State Counts Example	99
6.10 ASP Encoding of the Forward Order Strategy	106
6.11 A Sketch of a Concept Solution for the Goofy Gopher Problem (DP_3)....	110
7.1 Time Series Plots of Entities for Two Experts (from [108])	120
7.2 Comparison of Iterations among Entities for Two Experts (from [108])	121

Figure	Page
7.3 Comparing Trends in Using Abstraction for Two Classes of Students	123
7.4 Trends in Using Entity-Depth-Prevalence for Two Groups of Students	123
7.5 An Example of a P-Maps Sequence	126
8.1 Selected Decision Tree for Quantity	150
8.2 Selected Decision Tree for Variety	151
8.3 Selected Decision Tree for Novelty	153
8.4 Selected Decision Tree for Quality	154
9.1 Predicted Quantity in Backward Elimination for DP_4 Models	163
9.2 Predicted Quantity in Backward Elimination for DP_5 Models	164
9.3 Predicted Quality in Backward Elimination for DP_5 Models	164
9.4 Prediction Residuals for Different DP_4 Models of Quantity	165
9.5 Prediction Residuals for Different DP_4 Models of Quality.....	166
9.6 Prediction Residuals for Different DP_5 Models of Variety.....	166
10.1 Distribution of Students' Grades of PF Skills for a Design Task.....	179
10.2 Changes in Students' Problem Formulation Characteristics	181
11.1 Decreasing Variation in Variety (DP_4 To DP_5).....	193
11.2 Decreasing Mean and Variability of Average Novelty (DP_4 To DP_5) ...	194

CHAPTER 1

INTRODUCTION

1.1 Motivation

Problem formulation is an important step in the early stages of conceptual design which is believed to influence creative outcome, though it is an understudied subject [1]. A survey of the literature on empirical studies of designer thinking suggests that researchers have devoted considerable attention to ideation (generation of ideas or concept solutions), but not the pre-ideation stage (problem formulation) in conceptual design [2]. It should be noted that it is difficult to draw a clear line between problem formulation and ideation, as studies have shown that problem and solution co-evolve [3, 4]. However, it is not only useful to make a distinction between the two steps, but the effect of problem formulation on ideation should also be considered. As Harfield [1] put it:

“50 people starting from the same problem statement, come up with not 50 solutions to the same problem but 50 solutions to 50 different problems.”

In studying the effect of problem formulation on ideation, two key factors which differentiate designers are expertise and creativity. Many studies focus on the role of expertise, often in the form of comparing novices and experts [2]. Expertise is an apparent and explicit characteristic of a designer and can be directly queried, e.g. by counting years of experience in a field. Creativity, on the other hand, can be known indirectly. To determine whether a process is creative or not, it is appropriate to evaluate the outcome of the process with respect to a defined measure of creative outcome [5, 6].

Therefore, understanding creative problem formulation means to find out how differences among designers' problem formulations are related to their creative ideation. To reiterate, the following assumptions lead to the statement in the previous sentence:

- a) Creativity plays a central role in successful engineering design.
- b) Creativity can be evaluated by a measure of [ideation] outcome.
- c) Ideation might be affected by problem formulation.
- d) Problem formulation is an important yet understudied subject in design.

The main objective of this research becomes to find differences in designers' problem formulation. To that end, a model or structure is needed to see differences in characteristics of how different designers formulate design problems. One inspiring model which represents thinking about problems in a general way is Newell and Simon's Human Problem Solving [7]. However, the problems that they cover are well-defined problems such as chess or algebra. Chandrasekaran [8] performed a task analysis for design problem solving. He developed a list of subtasks and potential methods for each subtask to come up with a task structure. He considered design as a knowledge-based problem solving activity where designing is a search in a space of devices or components to a space of design specifications. Design problems are different from non-design problems, and the methods used and the results found for the latter cannot be generalized to the former. Goel and Pirolli [9] describe some of those differences. Even though Simon argues the possibility of finding structure in ill-structured problems [10], Dorst cautions about extending problem solving behavior of well-defined problems to ill-structured problems [11]. Design problems have other characteristics which must be considered in choosing an appropriate model for representing differences in designers'

problem formulation. In addition to being ill-structured (with conflicting goals, evident or explicit dependencies), design problems are ill-defined (with vague or incomplete goals), and dynamic (with changing requirements). Therefore, a representation of design problems in early stages of conceptual design, when the problem is formulated, should accommodate incomplete, conflicting, and changing problem definitions. At this stage, designers often reframe the problem space [12] and construct multiple representations of the problem [13]. Furthermore, a representation of problem definition should include elements of the solution space, since the problem and solution spaces co-evolve during design [3, 4].

Studies of problem formulation in design are scarce. More specifically, studies whose main objective is to understand and characterize problem formulation are rare and what are found in the literature are observations from studies with other objectives, often modeling the conceptual design process. These reasons motivate a dedicated study of understanding problem formulation with a higher level of detail, and an appropriate model which helps in showing the differences among designers in how they formulate design problems. Let us turn to the research questions which underlie this thesis.

1.2 Research questions and hypotheses

The main question to be investigated is that problem formulation plays a key role in creative design, and this role is not well understood, since dedicated studies to problem formulation in design are scarce and lack detail. There is a need for a structure or model to represent how designers formulate problems. A modeling framework based on a

predefined ontology is needed to represent problem formulation and study its relation to creative outcome. The main research questions then become as follows:

1. What model can be used to capture a designer's understanding of a design problem, and show individual differences in problem formulation?
2. How do more creative and/or experienced designers formulate design problems differently from less creative and/or novice designers? How can the differences be captured within the framework?
3. Can creative outcome be predicted from the way designers formulate problems?

The answer to the first question is required in order to reach the answer to the second research question which is to compare designers' formulations. The answer to the second research question is the models of the relations between problem formulation and ideation which provide the answer to the third research question.

The central hypothesis of this study is that problem formulation significantly affects creativity in design outcome, and creative and experienced designers formulate problems differently from non-creative and inexperienced designers do. A corollary to this hypothesis is that problem formulation characteristics which lead to more creative design can be taught to novices and the creative outcome can be predicted from problem formulation behavior. In addition, a few hypotheses can be formed based on observations from exploratory studies. Therefore, the following hypotheses will be tested:

H1_a) Novice designers follow a systematic order in expressing problem formulation while experts have a more opportunistic behavior.

H1_b) Experts find key issues early on during problem formulation while novices find more issues and later in the formulation process.

H2_a) Depth-first exploration of problem formulation entities leads to more creativity.

H2_b) Creativity can be improved in novice designers by teaching them characteristics of good problem formulation.

H3) Creativity in design outcome can be predicted with an acceptable degree of confidence from problem formulation behavior.

Hypotheses H1_a and H1_b are tested in an experiment which seeks the differences between experts and novices in problem formulation. Hypotheses H2_a and H2_b belong to an experiment which is about understanding the relation between problem formulation and creative outcome. Testing hypothesis H3 can be carried out with an experiment that examines if a model of the relation between problem formulation and creativity is generalizable.

1.3 Research tasks

To answer the research questions, three major tasks should be carried out:

1. Developing a modeling framework suitable for studying problem formulation.
2. Designing the experiments for the empirical study and collecting data.
3. Analyzing the data to test the hypotheses and propose new findings.

Each of these tasks includes a few steps. To achieve the first task an exploratory study can be conducted to observe how different designers formulate problems in a setting close to working on a real world design problem. The literature can also be reviewed on

the problem formulation process. Another step for developing the model is to choose an appropriate representation for modelling the process of problem formulation.

The second task involves recruiting participants with an appropriate representation of differences with regard to levels of experience and creativity, and choosing appropriate design problems which lead to variability in responses. Preparing the participants and controlling factors in the environment such as allowed response time are also parts of the second task.

The third task can be broken down into extracting information (intrinsic measures) from the data models, choosing an appropriate measure of creativity (extrinsic), and searching for patterns that reveal differences between more creative and less creative designers. Besides testing the stated hypotheses, new findings can be formed into new proposed hypotheses, and recommendations for problem formulation practices which lead designers to become more creative.

1.4 Guide to the dissertation

Throughout this research a broad range of the literature was surveyed in research in designer thinking. The fundamental themes were to learn about differences in the way designers approached design problems in early conceptual design, as well as pertinent representations and formalisms which facilitated modelling designer thinking. Chapter 2 covers the surveyed. This was a part of the first research task. Chapter 3 describes the steps taken towards developing a framework for representing problem formulation in design. It includes two exploratory studies for finding an appropriate structure to show differences in problem formulation data, and the desired specifications of a tentative

framework. Chapter 4 introduces the Problem Map (P-maps) ontological framework and the gaps in existing frameworks which necessitated the introduction of P-maps. The entities, relations and attributes of the P-maps modeling framework are explained. Pertinent modeling frameworks are compared to P-maps with respect to the stated specifications for a framework modeling problem formulation. Chapter 5 describes the Problem Formulation testbed which is built based on P-maps ontology to expedite data collection and analysis. Chapter 6 lays out the preliminaries of the conducted empirical study. It describes how the design problems and participants were selected for the study. Two types of problem formulation characteristics are defined. P-maps state measures are counts of entities or links at a certain time. Problem formulation strategies are defined as changes in an interval with certain conditions. Ideation metrics are explained as characteristics of creative outcome. The chapter ends with a summary of the design of experiments.

The following three chapters explain each of the three conducted experiments in detail. This includes the objective of each experiment in relation to the research questions and stated hypotheses, the collected data, the analysis methods used, and results and conclusions. Chapter 7 explains the first experiment which is to show differences between and within experts and novices. Chapter 8 describes the models of ideation with respect to problem formulation. 8.3.4 shows how the models found in the second experiment are used to predict creative outcome from problem formulation for other problems.

The level of detail which P-maps provide in characterizing problem formulation raises a few opportunities. Chapter 10 describes three potential applications of P-maps. They are

creating a test of design problem formulation skills, objective assessment of students' conceptual design skills throughout an engineering design course, and examining findings of previous researchers.

Chapter 11 concludes this dissertation by revisiting the research questions and hypotheses to examine how the findings answered them. Limitations of the study are discussed. Potentials for future research are also discussed including testing new hypotheses, creating a coaching system that aids novices in improving their problem formulation skills, and suggestions for overcoming some of the limitations faced during this research. The dissertation concludes with a list of original contributions and publications based on this research.

CHAPTER 2

LITERATURE REVIEW

In this chapter, relevant literature of conceptual design is reviewed. The main objective of this research is to understand the relation between problem formulation and creativity. Towards that goal, a model which is able to explain the relation needs to be created. A simplistic model of design is that a designer applies design knowledge (acquired internally or externally) to a design problem, following a process to come up with design solutions, see Figure 2.1. Different models of the design process add details to this simple version. Different studies focus on each of the elements in the simplistic model. Knowledge models and cognitive models focus on the designer. Expertise models focus on domain knowledge. Design theories, decision theories, and optimization models focus on the process. Artifact models, behavior models, and architecture models focus on design solutions. Affordances and emotional engineering attempt at modeling the user. Models can have different levels of abstraction. Design representations and how they transform are used in building models of the design process and solutions.

This view of design models can shape a basis for reviewing the literature on problem formulation as a part of the conceptual design process (which in turn is a step or sub-process of design). Three points can be taken from the simplistic model. One is that modeling design problems has received less attention in the literature. The other is that design representations are used in modeling both the design process and the design outcome. The third point is that the design process is a link between design problems and solutions; in other words models of the design process (formulation) and outcome may be worked backward to a model of the problem. I shall also add that the widely accepted

notion of the co-evolutions of problem and solution spaces [3, 4] suggest that creating a model of design problems cannot be done without considering elements of design solutions.

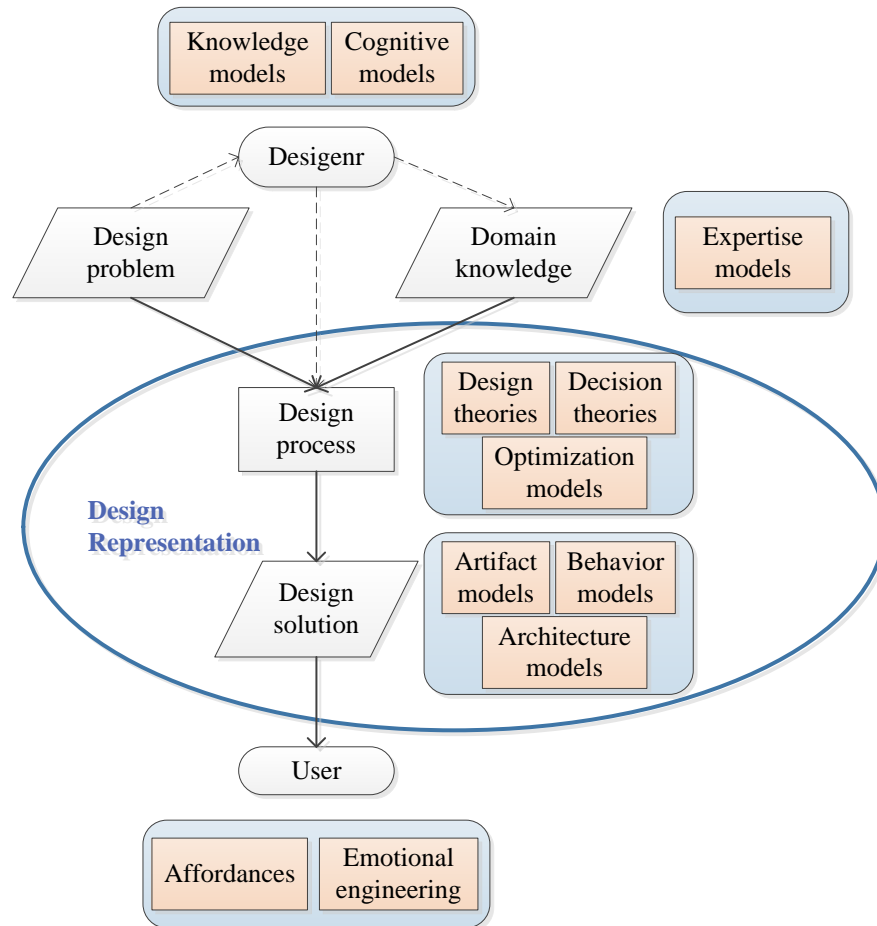


Figure 2.1 A simplistic model of design (adapted from [14])

For these reasons I will review two major themes in the literature. One is about the formulation process, and the other is about representation models. More specifically, one focuses on the literature around how [differently] designers think during conceptual design, i.e., how they approach a design problem, frame and reframe the problem, and attempt to solve the problem by generating ideas. The other major theme in the literature

review will be on relevant representation models of designer thinking which underlies the methodology pertinent to what is proposed in this thesis: the application of an ontological framework for an empirical study of designers' problem formulation. The literature is searched for similar frameworks and ontologies that have been implemented in design studies, relevant representations in design, as well as inspiring formalisms in other research areas such as knowledge representation in education.

2.1 The process of problem formulation in design

Little research has been conducted to understand how problem formulation affects creative outcome in engineering design. Review of the design literature reveals a few studies that have focused on representing the problem and the solution spaces, as well as some on the process of problem formulation. This section starts with a review of problem formulation in design to highlight the types of data fragments that are present in designers' problem formulation, and the differences that should be looked for among designers. Creative and experienced designers approach design problems differently and adopt different strategies from non-creative and novice designers. Therefore, the literature in this section is centered on three close threads: a) processes, methods and best practices; b) strategies; and c) differences between novices and experts.

2.1.1 Processes, methods and best practices

Two of the earliest studies of mechanical designers are Ullman *et al.* [33] and Waldron and Waldron [16]. Both studies were interested in developing a general model of the mechanical design process, and quantifiable measures for its assessment. Ullman *et al.* asked individuals to work on two simple problems while Waldron and Waldron asked

design teams to work on a vehicle with complex mechanisms. Ullman *et al.* [33] defined the Task-Episode-Accumulation descriptive model. They broke down the transcript into units that could be classified as operations which alter the design state. This state-operator modeling will be discussed more extensively in the next section on representations and formalisms. Waldron and Waldron [16] discovered extensive use of biological analogies, experts' bias towards first concepts, and experts' opportunistic approach of quickly identifying and devoting initial focus towards the most critical parts of a design.

Protocol studies focusing on the conceptual design process has shown a few characteristics of problem formulation. Designers prefer to treat problems as ill-defined [17, 18]. Atman et al [15] state that senior undergrad design students produce higher quality designs by gathering more information early, considering more alternative solutions, and moving more frequently between design steps. Eisentraut [20], however, maintains that such behavior relates to different styles of problem solving, which are independent of the situation of the design task at hand.

Unlike well-defined problems, design problems continue to evolve throughout the problem solving process. It is suggested that recognition of partial structures in the problem space, shape the structure of the solution space [3, 4]. Cross and Cross [18] claim that creative designers, holding experience of previous solutions at the back of their minds, use first principles as stimuli to build bridges between problem and solution space through key concepts. Harfield [1] claims that designers need 'proto-solutions' to compare the goal and the problem state, and that naive designers make fixed assumptions while creative designers question requirements.

A major line of investigation is related to blocks and resolution of impasses in design creativity [10, 16–18, 26]. Dorst *et al.* [4] has studied how co-evolution of the problem and the solution affects creativity. This aspect has also been corroborated by Kim and Maher [26] and Lemons *et al.* [27]. Gero *et al.* [28] has studied the effect of “structuredness” of three ideation methods on design cognition to find that the more structured a method is, the more designers tend to focus on design goals and requirements. Similarly, Valkenburg and Dorst [12] suggest that a more successful design team frames a design problem more frequently than an unsuccessful one. Christiaans and Dorst [29] have shown that designers who spend more time on problem definition are more likely to come up with better designs. They also have found that more successful designers concentrate on progressing to solution generation and building up an image of the problem. Fricke [30] suggests that successful designers ask sets of questions related to problem structure, and clarify requirements, functions, and technical characteristics representing the problem structure.

In addition to the observations that describe the design process, there are some prescriptive models of engineering design that offer different methods and checklists for every step of the design process. The Association of German Engineers (VDI) systematized engineering design through a series of guidelines, of which VDI 2220 and VDI 2221 relate to the earlier stages of design. More notably, the systematic approach of Pahl and Beitz [31] introduced a checklist for developing requirements with a list of examples for geometry, material, ergonomics, assembly, etc., spanning the product life-cycle. Requirements are not only specified individually, but also lead to other

requirements, often in a parent-child relation. Developing an objective tree is a common method of eliciting new requirements and determining how they should be synthesized.

Another well-established aspect of problem formulation is the development of function decompositions. Similar to objective trees, function trees are developed to find out what different parts of the design should do to achieve its main purpose. Functions are decomposed into sub-functions until referring to a specific solution becomes inevitable, and no more abstract functions can be defined. Otto and Wood recommend functional decomposition as a useful method in product design [32]. They have created a collection of function decompositions by reverse engineering some consumer products. This raises the question that if the method is only appropriate for redesigns and not coming up with novel designs. Creating alternatives (disjunctive decompositions) may resolve this shortcoming.

There are other methods which have been used in early stages of design for problem definition. The QFD method [33] relates and quantifies customer needs in relation to design parameters. However, prior knowledge about those parameters is central to the application of these methods. Such knowledge is often absent in the fuzzy front end of formulating design problems which involve new and novel products. Therefore, a well-established method such as QFD which deals with evolutionary development processes of mature products is not applicable.

2.1.2 Strategies

Besides studies of processes in a general way, more specific strategies that are adopted in problem formulation should also be considered. Some of these studies define strategies

in a broad way. Kruger and Cross [34] categorize designers into problem-driven and solution-driven. Gero and Mc Neill [35] classify the different strategies that designers adopt into micro strategies (analysis, proposition, and making explicit references), and macro strategies (top-down, bottom-up, decomposition, opportunistic, and backtracking). As stated in the previous chapter, an influencing strategic behavior in conceptual design is abstraction. Ward, Patterson, and Sifonis [21] have conducted experiments to investigate the role of abstraction in creative ideation. By actively instructing the participants to formulate the given task in either very specific or more abstract ways, they have found that the latter instructions led to more novel ideas. Ball, Ormerod, and Morley [16] have found that experts lean on experiential abstract knowledge while novices rely on case-driven analogies, mainly driven by surface-level cues.

Problem decomposition is another designerly behavior that can affect the outcome of conceptual design. Liikanen and Perttula [14] have analyzed the prevalence of explicit and implicit problem decomposition modes through a protocol study involving 16 senior students of mechanical engineering. In this context, explicit decomposition means deliberate creation of a decomposition, e.g., creating a function structure as some design textbooks advocate. They have found that the subjects implicitly employ top-down problem decomposition while explicit decomposition is rarely used and often does not foster creativity. In contrast, Ho [14] have found that expert designers are more likely to utilize explicit problem decomposition, leading to more creative ideas. One can infer a depth-first exploration from this observation, though Ho's study involves one sophomore industrial design student as the novice and one graduate with half a year of professional experience as the expert. Contrarily, Ball et al. [12] have conducted a protocol study

where they have observed experts use more breadth-first search while novices use depth-first search in ideation. However, they also report that experts utilize a strategic knowledge about how to conduct the design process effectively when they face impasses, by switching from a predominantly breadth-first mode of problem solving to an opportunistic depth-first mode. In another protocol study with three subjects Cai, Do, and Zimring [13] have found no relation between creative outcome and depth vs. breadth exploration of the design space.

2.1.3 Differences between novices and experts

In addition to general observations about designers, protocol analysis has led to observations about major differences between novice and expert designers and/or more successful and less successful designers. Kavakli *et al.* [42] have found that experts' cognitive actions are organized while novices have many concurrent actions that are hard to categorize. Ahmed and Christensen state that experts tend to use analogies for predicting component behaviors and problem identification whereas novices tend to transfer geometric properties with evaluating the appropriateness of analogies [43]. Comparing freshman and senior engineering design students, Atman *et al.* [44] have found that seniors produce higher quality solutions, spend more time solving the problem, consider more alternative solutions and make more transitions between design steps than the freshmen.

2.2 Representation frameworks and formalisms

This section of the literature review focuses on pertinent frameworks and representations in design and other inspiring fields of research such as knowledge

representation in education. Frameworks which provide a computational means for contrasting characteristics of different designers are of interest, but ones that are appropriate in conceptual design. Therefore, conventional CAD models which represent models of detailed embodiments are not in the scope of this review. Three threads are looked into: a) design representations, b) ontologies, and c) computer formalisms.

2.2.1 Design representations

A few researchers have developed models for representing the structure of design problems. Maher et al. [3] have linked problem definition states to solutions in an abstract way. Goldschmidt [45] has attempted capturing the indeterministic nature of design by providing multiple representations of figural-conceptual modes—with their equivalent external representations, i.e., sketches and verbalizations. In her node-link representation, she equates states and operators in problem solving with nodes, and their sequences with links. Later Goldschmidt and Tatsa [46], use linkographs to show that intensive interlinking breeds more creative designs.

Cai, Do, and Zimring [41] have developed an extension of linkography in addition to a distance graph to investigate design patterns among designers of different expertise levels and exposure to different stimuli. They modify the definition of links based on lateral transformation and vertical transformation to represent both the breadth and the depth of the problem space explored in design. In lateral transformation the movement is from one idea to an alternative. In a vertical transformation the move is from one idea to a more detailed or elaborated version of the same idea. They report that the more creative the

design is, the higher number of alternatives and the more chunks and webs are displayed in their representation, the more extended the linkograph.

In a different application of linkography in finding patterns in conceptual design, Kan and Gero [47] conduct protocol studies to acquire information from linkographs. They define two methods to abstract information from the linkographs: one based on clustering, and one based on Shannon's entropy measure. They state that cluster analysis is able to group the linkographs into meaningful clusters, while entropy measures the opportunities for idea development.

In characterizing the differences between design and non-design problems, Goel and Pirolli [9] have come up with a Task-Operator-Phase model, inspired by information-processing theory of human problem solving [7]. Similarly, the Task-Episode-Accumulation (TEA) model of Ullman *et al.* [33] has been one of the pioneers not only in adopting protocol analysis for studying conceptual design, but also in describing the design process through a state-operator model. The TEA model defines the design process as applying a sequence of operators (such as select, simulate, compare, reject, refine) during episodes (such as plan, specify, verify) to achieve a goal in a design task (such as conceptual or detail design).

Some studies have proposed and utilized specific modeling frameworks similarly to the general approach taken to this thesis, though the motivation for a new framework has been highlighted in Chapter 1. An established framework in representing design thinking is Gero's Function-Behavior-Structure [48]. Gero [48] has defined activities in the design process in terms of transformations from one of the three domains of Function, Behavior, or Structure to another, considering a difference between expected and actual behavior,

see Figure 2.2. In this model, the purpose of designing is to transform function (F) into a design description (D), though this cannot be done directly without other transformations. For example, inferring expected behaviors from functions is considered formulation (process 1). Three processes are described as reformulation of structure (process 6), expected behavior (process 7), and function (process 8). All three reformulations are transformations from structure which represents artifacts and their relationships.

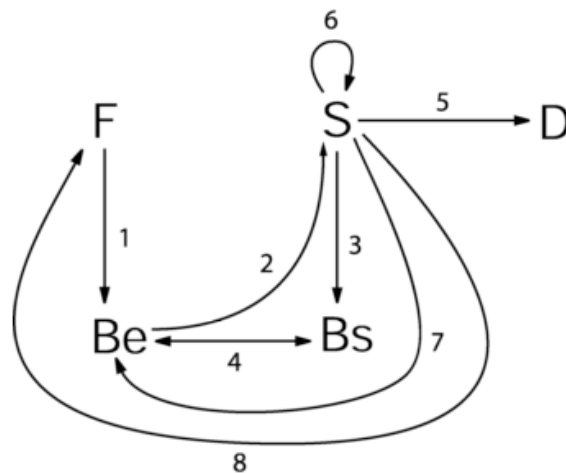


Figure 2.2 A model of activities in design in the F-B-S framework (from [48])

Gero and Kannengieser [49] have taken into account the dynamic character of design by considering the notion of situatedness. F-B-S has been used in modeling the design process [48], as a coding schema in protocol analysis [35, 50], and for design automation [51]. Even though F-B-S has been used as a predefined coding schema in protocol analysis [50], it has not been used as a computational framework for searching for strategies because, as Gero and Kannengieser contend [52], F-B-S is a high-level model.

There are similar models to F-B-S, which have been developed independently and with different purposes. Prior to Gero, Chandrasekaran had proposed Functional Representation (FR), initially as a knowledge representation for an expert system which

generated relationships (in addition to compiling stored relations) between functions and structures (Sembugamoorthy and Chandrasekaran [53]). FR was a language which described the function of an artifact in terms of causal processes in order to simulate, diagnose or explain how the artifact works. A retrospective account of FR and its applications can be found in [54].

Umeda *et al.* [55] proposed the Function behavior-state diagram; see Figure 2.3. Their main goal was to clarify the definitions of function and behavior, and to incorporate a hierarchical structure for functions. They substitute structure with state (as a state of a structure in an instant) and argue that the distinction between the two depends on time which is irrelevant to an instantaneous representation. They define function as an image of a behavior abstracted by humans. Therefore, functions and their relations to behaviors are considered subjective elements of a design object while behaviors and states are objective or physical. The function hierarchy is separate from the representation of behaviors and states. Only functions can form hierarchies and each function can be related to a Behavior-State description. Umeda *et al.* have used their model in developing the FBS modeler computer tool to support functional design [56] and a method for extending the life-cycle of products by finding possible changes to functions that can be adapted to with minimal structural changes [57].

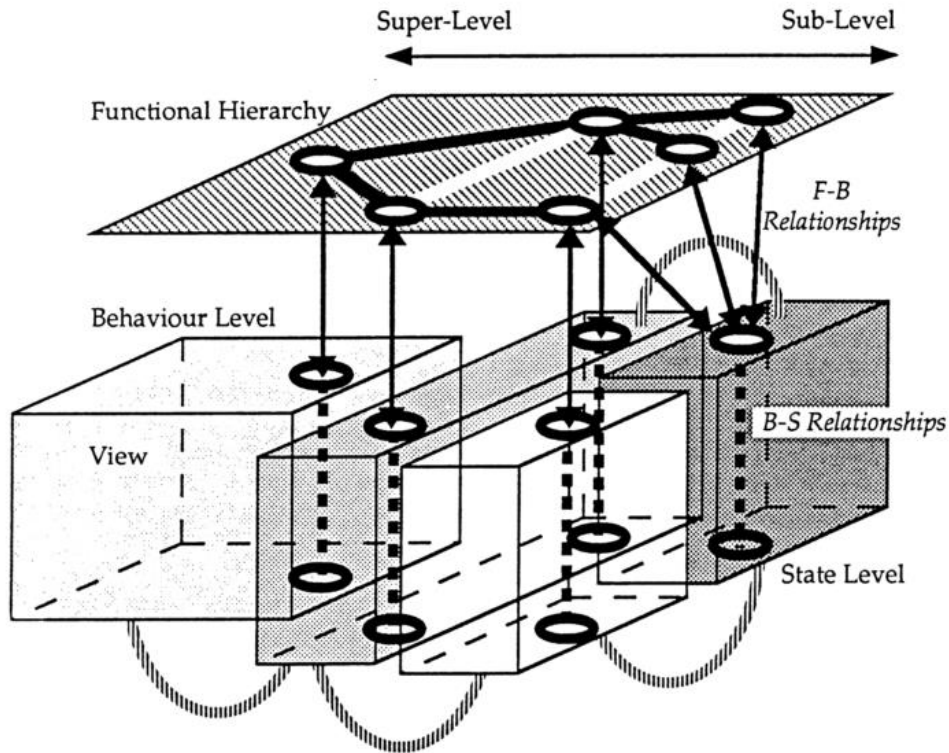


Figure 2.3 Function-behavior-state diagram (from [55])

Goel, Rugaber, and Vattam [58] have developed the Structure-Behavior- Function (S-B-F) modeling language for a teleological description of complex systems. In this language, structure, behavior, and function are represented in terms of components and their connections, transitions among a sequence of states, and pre- and post-conditions respectively. The syntax is similar to notations that are used to represent production rules. The model is a top-down description scheme, in which each fragment of the model is defined by a lower level fragment. At the top, there is an instance of S-B-F, while at the bottom there are building block fragments such as strings and integers. For example, an *element* (a component in a structure model) is defined by an integer *Id*, a string *name*, a string *description*, an optional set (can have zero number of fragments) of *property*, and

an integer *subelement Id*. The different variants of the F-B-S family seem to have a common objective which is modeling existing designs. In a broad sense, they are mainly product models.

More recently, Helms and Goel [59] have proposed the Four-Box method with the objective of helping students to formulate problems and evaluate analogies inspired by biological analogues. Using grounded theory methodology, they have created a structured representation for biologically-inspired design (BID) which has served as a coding schema for mapping problem specifications to BID analogues; Figure 2.4 shows the Four-box diagram.

Operational Environment	Function
Specifications	Performance Criteria

Figure 2.4 The Four-Box diagram (from [59])

They propose four entities to describe a problem in a way that can be searched for and compared to a database of existing biological analogues defined along the four entities with varying degrees of similarity (defined as ‘same’, ‘similar’, and ‘different’). They are *Function*, *Operational environment*, *Constraints/specifications*, and *Performance criteria*. For example, in designing a light post, a Saguaro Cactus as the chosen analogue has the following characteristics in common with the light post: ‘outdoors’ as the ‘same’ *operational environment*, ‘collect light’ as a ‘similar’ *function* to ‘project light’ in a light post, ‘bright’ as a *specification* of a light post ‘different’ from the analogue, and

‘withstand 70 mph’ as a ‘similar’ *performance criteria* to the analogue, see Figure 2.5 for a detailed comparison.

Light Post		Saguaro Cactus	
Operational Environment		Operational Environment	
Outdoors	SAME	Outdoors	
Wind	SAME	Wind	
Precipitation	SAME	Desert conditions	
Temperature Variation	DIFFERENT	Large temperature variation	
Vibrations from wind and road	SIMILAR	Occasionally deals with these	
Functions		Functions	
Project Light	SIMILAR	Collect light	
Elevation of light source	SAME	Protect water from scavengers	
Vibration reduction	SIMILAR	Resist breakage	
Increase dampening	SAME	Dampen wind with spines	
Specifications		Specifications	
30' tall	SIMILAR	Up to 45 feet tall	
Bright	DIFFERENT	No light production	
Requires mast arm	SAME	Branches	
Decrease materials	SAME	Optimizes energy expenditure to support self	
Easy installation	SIMILAR	Optimize energy used during growth	
Low cost	SAME	Optimize total energy use	
Performance Criteria		Performance Criteria	
Withstand up to 5g's	DIFFERENT		
Withstand vortex shedding	SAME	Withstand vortex shedding	
Withstand 70 mph winds	SAME	Withstand up to 85mph winds	
Infinite lifespan	SIMILAR	May live to 200 years	

Figure 2.5 An example of evaluating a BID analogue in the Four-Box (from [59])

Though the proposed structure is useful in describing a problem in such a way that can facilitate a search for BID analogues, it arguably has overlaps in the definition of specifications and performance criteria. In addition, it still is based on human judgment for determining the relevance of an attribute in the description of a problem to the predefined analogues. A compounding problem is that the database of existing biological

analogues has been also developed based on human judgment. It should be noted that the empirical study conducted by Helms and Goel [59] has shown around 80% accuracy in generating a problem definition (compared to normative problem definitions created by the authors for 15 design problems) among about 50 students of a BID course with diverse backgrounds. However, it is not clear how the norms are generated to establish the accuracy measure. The evaluation method is also based on a protocol analysis where the students' code their concepts post-generation to one of the four entity types which is compared to a judge's coding. There is not a clear connection between the level of agreement between the two coders and the definition of accuracy:

“The concepts are also assigned a code based on the section in which it was placed by the student. The rater-assigned code concept is compared to the student-assigned code, and is evaluated in as either “agrees” or “disagrees.” The degree to which, for any category the two codes agree may be expressed as a percentage of total concepts in agreement over the total concepts encoded. Accuracy is compared between-groups for differences among: gender, major, year (2011 or 2012). Accuracy differences are also compared among the four conceptual types.”

Besides developing modeling frameworks that can be used commonly in studying different aspects of design cognition, others have tried to employ standard modeling languages. Wölkl and Shea [60] have used SysML in modeling conceptual design. They follow the prescribed systematic engineering approach by Pahl and Beitz [31] and the German standard VDI 2221. They propose creating new specification with Requirement diagram, describing functions with Use Case diagram and Activity diagram, and

allocating working principles with Block diagram. Using such a standard language makes it easier to integrate the often non-geometrical data of conceptual design with later stages of product development. However, Wölkl and Shea [60] concede that the representation is not compact from usability viewpoint, and multiple (and separate) diagrams are required to represent different aspects of the designs. This makes it less likely to see the problem in context, or boost creative ideas which often arise from seeing the inter-connections of concepts [6].

2.2.2 Ontologies

A different approach towards implementing design representations is to go beyond how the structures of the representations look and focus on what the meanings are within a structure. That is to understand the role and the application of ontologies in design. There is not a clear definition of what an ontology is in a design research, since historically it has been a concept in philosophy. Ontologies are pertinent to problem definition because they intimately involve language (textual/verbal mode/representation). Conceptual design does not merely involve form which is supposed to be more effectively expressed by sketches [61]. In early stages of problem formulation, prior to expressing any forms or embodiments, words can have a higher efficiency of describing abstract design thoughts [62, 63].

Another issue that involves ontologies is search through words. Regardless of growing computing power, search results can become overwhelming for the user to filter through when employing knowledge bases without a proper structure that maps onto the domain at hand. Most knowledge bases such as WordNet [64] have ontologies more suited

towards common sense knowledge, not design or engineering. There is a need for an ontology specific to design but also not limited to technical terms which can be found in some design repositories such as Bohm *et al.* [65], since the fuzzy front end of the early stages of conceptual design, especially for novel designs, is often described in a less formal language.

A conventional definition of an ontology is a taxonomy plus inter-category relations, i.e., a taxonomic structure that represents knowledge with defined relation types among the categories of the taxonomy. Uschold [66] defines an ontology in the following:

“An ONTOLOGY may take a variety of forms, but necessarily it will include a vocabulary of terms, and some specification of their meaning. This includes definitions and an indication of how concepts are inter-related which collectively impose a structure on the DOMAIN and constrain the possible interpretations of terms.”

In engineering design research, different ontologies have been proposed with either generic or specific scopes of applications. Sim and Duffy [67] have defined a generic ontology of engineering design activities by creating a structure for a set of steps in a general design process, and for design generation, evaluation, and management activities. Each step includes four elements which may be related in a specific way: the goal of the design activity (G_d), the input knowledge (I_k), the output knowledge (O_k), and the knowledge change. For example, for the design activity of abstracting, the four mentioned are as the following respectively: to simplify the complexity of the design object (G_d); types of abstraction (I_k); appropriate abstractions of design object, e.g., sketches (O_k); and knowledge abstractions that depict useful relationships of the evolving

design concept. The objective of such ontology is creating a coherent interpretation of definitions of the activities in order to have more effective design support.

A less generic (in terms of structure rather than content) ontology is the application of the SAPPPhIRE model by Srinivasan and Chakrabarti [68] which has been developed to explain the knowledge of biological and artificial systems in design problems with a generic causal behavioral model. The entities in SAPPPhIRE are *State* (S), *Action* (A), *Part* (P), *physical Phenomena* (Ph), *Input* (I), *oRgan* (R), and *Effect* (E). Based on this representation, Srinivasan *et al.* [69] have developed an ontology by building clusters of nouns, verbs, adjectives, adverbs and mathematical equations from earlier work with the SAPPPhIRE model. They have also compared their ontology to others.

Another specific ontology is the reconciled function basis by Hirtz *et al.* [70] where different researchers from academia and NIST contributed to a vocabulary of abstract sub-functions, in order to make functional decomposition more methodical. The objective was to form a set of functions that would ideally lead to a minimal set of terms that did not overlap, and yet provided complete coverage of designed products. Different function bases were combined to reach a unified vocabulary for a standardized development of function trees.

2.2.3 Computer formalisms

So far, the review of the literature on representation models has focused on engineering design. There are inspiring formalisms in software engineering and computer science that should be mentioned for two reasons: such formalisms have been used for representing knowledge, and thus [design] thinking (representation aspect); they will be

pertinent to automating analyses of design thinking data (computation aspect). UML models (which are the basis for SysML) are good at representing a specific class of problems, often related to a specific class of artifacts or systems. Since sub-classes inherit the attributes and the functions of their super-classes, UML models excel at compactly defining classes of objects because they avoid redundancies. Database models such as Entity Relationship Diagrams (ERD) are one means to organize data and are more concerned about compact relations in order to respond quickly to queries. In this research, expressiveness is a more important objective than compactness, while ERD's are concerned with the latter.

Concept map [71] is another representations that has been used in education as means of providing students with an easy and intuitive way to document and explain taught lessons. This provides concept maps with insight into the systems they design [72]. Novak and Cañas [73] have proposed the use of concept maps to identify changes in students' understanding over time. Additionally, concept maps have been used to understand the differences between the knowledge of experts and novices. The main advantage is the ability to accommodate fine levels of granularity. Even though concept maps have nodes and labeled links, and can represent hierarchies, they are still relatively unstructured. There is no standard way or ontology and one can label data fragments in any way. This becomes a major shortcoming, especially when one wants to compare different instances of the problem formulation over time or to compare models of different designers. Figure 2.6 exemplifies a concept map of a problem formulation process of designing a water sampling device.

Concept maps have been used with some modifications in research in design. One example is Oxman's Think maps [72]. Differently from what will be shown in the next chapter, the medium in Think maps is a non-hierarchical concept map, and the objective for the ontology is teaching domain knowledge (comparison of a student's map to that of a teacher or norm). The similarities to the P-maps ontology are using a computational framework (method), and educating students by comparing them to a normative knowledge structure (application).

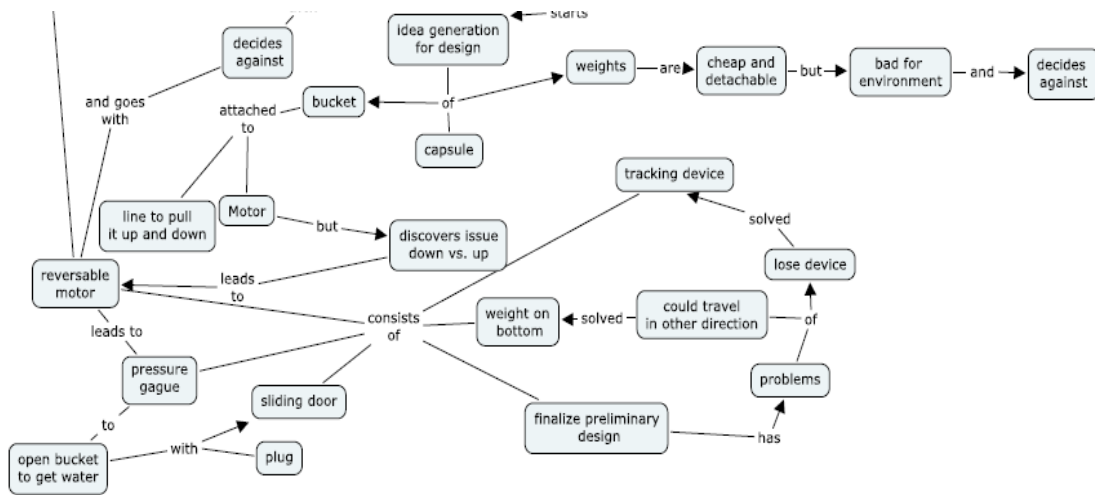


Figure 2.6 A concept map of formulating a design problem

Semantic networks [74] are a type of graphical network that relate conceptual nodes with binary links. They have been used to represent the meaning of sentences in natural language processing. Nodes are used for representing concepts and links for the types of relationships among them. This is a graphical representation of some static situation, e.g., a person's mental state. Concepts are usually organized in a taxonomic hierarchy and

often rely on the use of inheritance [75]. Semantic networks struggle to represent disjunction [75], which is important in representing design problem formulation.

There have also been efforts in combining the different representations and search methods that were described above. An example is Hao et al. [76] where they extend previous research on concept map assessment, to develop an evaluation metric in order to predict individuals' problem-solving performance. They propose their EntropyAvg novelty metric based on Shannon's entropy in information theory. They have conducted a controlled experiment where they find a strong correlation between individuals' problem-solving performance and their EntropyAvg measure.

To summarize, the literature review covered previous studies in understanding problem formulation, in addition to some of the representations that have been developed for studying design thinking and modeling design processes. A few formalisms that might be used in representing or building a computational model of design thinking were also described. Studies of problem formulation have been fragmented, and representation models that have been proposed in studying design cognition, though have led to interesting findings, do not have the necessary level of detail for studying problem formulation. Therefore, there was a need for a new modeling framework that was fine-grained, and incorporated formalisms that facilitated showing differences among designers' problem formulation. The next chapter explains the process of getting to that new modeling framework.

CHAPTER 3

TOWARDS A STRUCTURED REPRESENTATION

The main motivation behind this research is to discover the influence of design problem formulation on creative outcome. To study problem formulation in design, there is a need for a structure to represent how designers formulate problems. Review of the literature showed pertinent representations, ontologies, and modeling frameworks but it also discussed the need for a new framework. This chapter describes how this framework was created.

Development of the framework required three steps. First, two exploratory studies using protocol analysis were conducted to find problem formulation entities and an appropriate way to represent them. The second step involved expanding the search in the literature to create an exhaustive list of relevant problem formulation entities. In the last step, the entities were synthesized into a smaller set. Similar entities were combined, and the definition of the finally selected entities was broadened to cover similar entities as much as possible. I should add that exploration, refinement and synthesis were not entirely separate. This process was carried out spirally and on a micro-level throughout the development of the framework.

The reason why similar entities were combined was that the target ontology should be easy to learn and remember for prospective users of the ontological framework. Therefore, compactness is a desired feature for the ontological framework. Other specifications of an appropriate framework arose during the exploratory studies. They will be discussed in the last section of this chapter.

3.1 Initial modeling structure

The first exploratory study was carried out to identify problem definition terminology, and also as a first attempt to come up with a structure for representing problem formulation data. To meet this objective protocols collected from two groups of undergraduate designers and an expert were analyzed [77]. This section describes this exploratory study including the design task, the data collection settings, the protocol coding process, and the modeling structure that emerged.

The task was designing a remotely-controlled model plane for a multi-objective competition where speed of the plane and its load carrying capacity would be tested with different scoring weights for each mission. The route followed an oval course with a 360 degree loop as seen in Figure 3.1. The problem was taken from the AIAA¹ Design/Build/Fly annual competition. The problem statement had restrictions on materials, motors, and propellers that could be used. There were also other constraints: the plane had to be hand launched, battery powered, and self-landing.

Protocols were collected from an expert designer with more than 16 years of experience in building about 100 model planes, and two groups of 4 senior undergrad students. Design sessions were recorded by two video cameras. The participants were told that they had an hour to work on the problem, though there was no pressure on keeping the duration exact. One group sat about forty minutes while the other group stayed about an hour and a half. They were asked to verbalize their thoughts without considering whether what they were saying would make sense to someone else. They

¹ The American Institute of Aeronautics and Astronautics

were allowed to write and/or sketch as desired. Throughout the session an experimenter was present in the room but out of the participant's sight. The experimenter's role was to ensure that the session was recorded and to prompt the participant if they fell silent. Audio and video of the session was collected and later transcribed.

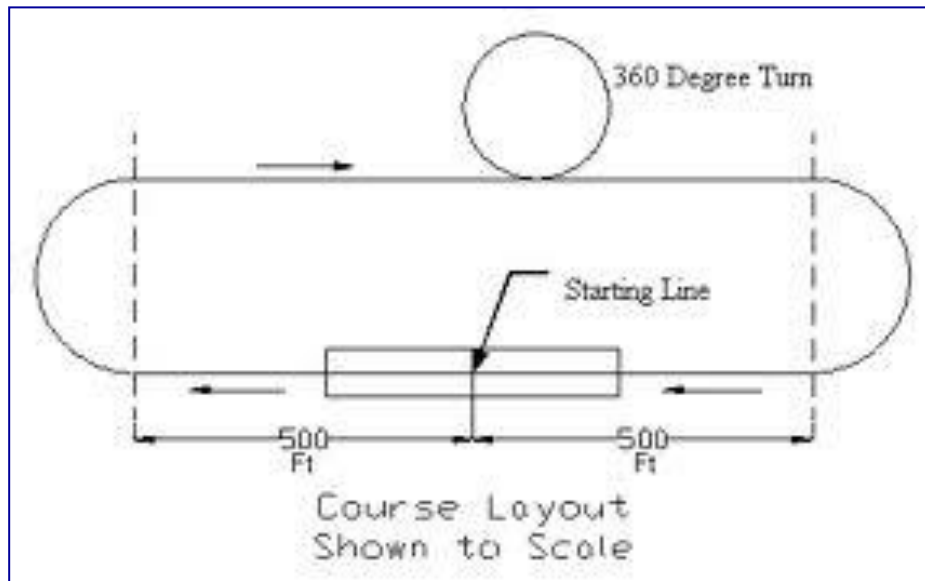


Figure 3.1 The design task of the first exploratory protocol study

The coding process was as follows:

- 1- The protocols were divided into short segments in such a way that each segment would be an answer to one of three high level questions:
 - a. What does the designer discover?
 - b. What does the designer exploit?
 - c. How does the designer treat or approach the problem?
- 2- Each segment would then be given a more specific label (an entity or class) such as rule, or insight.

- 3- If a segment could be labeled with one of the existing labels it would be given that label, otherwise a new label would be created.

This coding process might seem to be arbitrary but it is common in protocol analysis to develop the coding schema as one goes through the data. Using predefined coding schemas are the exception, not the norm. One example of using a predefined coding schema is done by Pourmohamadi [50] using Gero's F-B-S. There are no standard ways of coding protocols [2].

The results of the initial coding are exemplified in Table 3.1. As I explained in the introduction of this chapter, exploring the literature and refining the ontology was an ongoing process in developing the framework. Some of the initial entities shown in Table 3.1 were dropped, new entities were added, and a new structure was adopted to represent the relations among the entities. There were two reasons behind this restructuring: the focus should be on formulation, not idea generation; a representation that showed a state at a moment rather than a process was preferred (a process would be represented by a set of states or snapshots). Thus an entity such as *decision* was removed. Decisions could be shown by comparing two snapshots of the process at different times. Entities which seemed redundant or vague were also eliminated; a *perception* would be implied in other entities.

Table 3.1 Coded segments from the first protocol analysis

Question	Entity	Example of a segment
What the designer discovers	Function	<i>"...it has to land, it can't sustain too much damage..."</i>
	Constraint	<i>"...at what velocity it needs to get in the air... if somebody can throw it like that..."</i>
	System hierarchy	<i>"...we need to get a basic design of the whole thing..."</i>
	Parameter	<i>"...its called the aspect ratio, 0.4 is a good number..."</i>
What the designer exploits	Domain knowledge	<i>"...that's why you throw it up... so the acceleration back down gives us a boost..."</i>
	Physical rules of behavior	<i>"...the smaller surface areas at the front, the better for the aircraft to fly; there is minimum drag..."</i>
	Relations	<i>"...what's affecting you the most is surface area, and that's for drag..."</i>
	Insights	<i>"...maybe our plane doesn't fly that high and this [variable] in the formula could be one..."</i>
How the designer treats the problem	Priorities	<i>"...that's a good goal, with the weights that we have and the power system [selected] well be able to determine the velocity required to get the lift needed..."</i>
	Perceptions	<i>"... we can have two pieces of fuselage if we want..."</i>
	Decisions	<i>"... we have to decide for pusher or puller..."</i>

The updated ontology and the modeling structure that emerged from the restructuring can be seen in Figure 3.2. Three groups were similar to the Function-Behavior-Structure

model of Gero [48]. The group *Structure* defined the solution structure in a hierarchical system with entities *component* and *parameter* corresponding to different levels of detail. The *trade-off* entity set relationships among parameters, often when having opposite effects. The group *Usage* had entities which determined what constraints should be considered in realizing the problem. This was not limited to the constraints that were directly imposed by the design brief but also what the use environment required. The group *Concerns* related to the *questions* that were raised, *issues* that were deemed to be pivotal in the feasibility of the solution and the *priorities* that were set during designing. This group could represent why decisions were made and what insights occurred to the designer; decisions and insights were omitted from the initial model. Finally, the group *Knowledge* corresponded to the application knowledge [34] in design problem solving. It referred to what was required in domain knowledge or what inferences were made from experiential knowledge. Relations which were found among segments in the protocol are shown with the lines in Figure 3.2.

The coded segments were assigned new labels based on the new ontology. I give examples from one novice group and the expert, since the novices had fairly similar problem formulations. The novice group considered the trade-off between the weight and the speed and recognized how different functions (thrust and lift) and their behaviors were related through a physical rule (the Bernoulli rule); they said "... with the weights that we have and the power system [selected] we'll be able to determine the velocity required to get the lift needed ...".

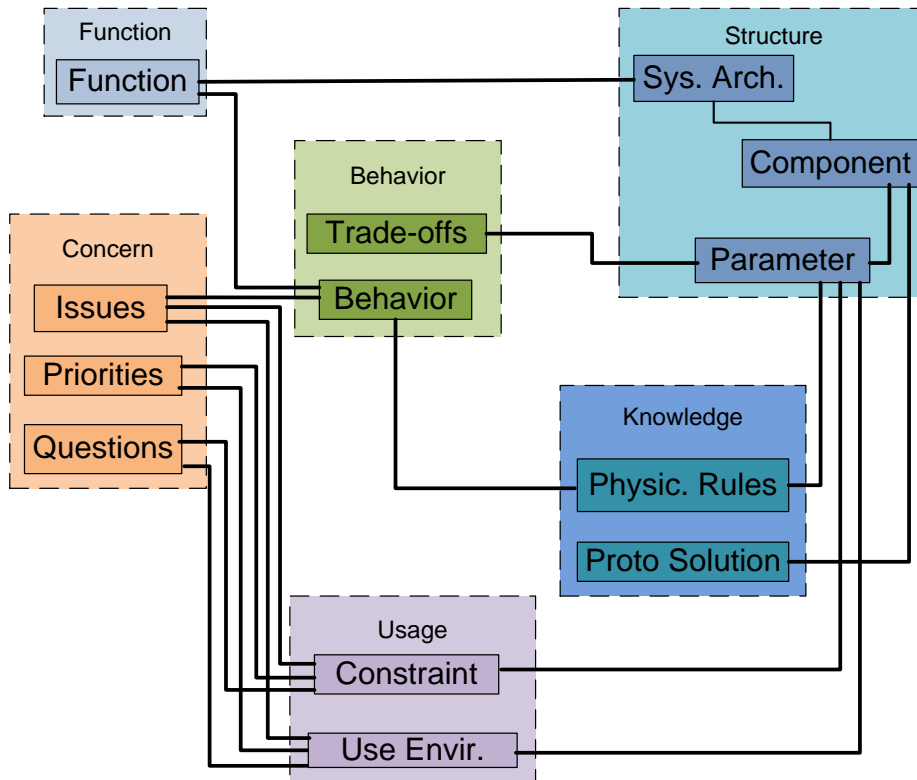


Figure 3.2 The problem formulation ontology from the first exploratory study

Knowledge about key rules differentiated the expert in implicitly drawing relations among many entities. He quickly pointed out that “... the ratio of the wing surface to plane speed should be in this area ...” referring to a ‘load-speed’ chart in aircraft design. Such insights prompted issues which were mostly neglected by the novice group. For example the expert mentioned that “... very rarely is it possible to design an aircraft whose payload is equal or greater than its weight ...” and concluded that “... we’re gonna have to design for high lift ...”.

The next step was to create a representation which would make it easier to highlight the differences between the expert and the novice group and changes in problem formulation in time. Once the segments were assigned to one of the entities in the

ontology, they were given a distinct name or short phrase and put in a box under the entity. For example the segment "... at what velocity it needs to get in the air... if somebody can throw it like that ..." was given the name "Hand-launch" and put under *issues*. Relations were drawn similarly. The segment "... with the weights that we have and the power system [selected] we'll be able to determine the velocity required to get the lift needed ..." implied relations among *parameters* "battery weight" and "wing weight", the *physical rule* " $\frac{1}{2}v^2 + \rho gh = c$ ", and the *function* "lift".

To show change, different snapshots could be created were each snapshot had all the coded segments up to the time of the snapshot. Figure 3.3, Figure 3.4, and Figure 3.5 show three snapshots respectively: the novice group halfway through their session; the novices at the end of the second; and the expert at the end of the design session. For simplicity and easier comparison all segments are not shown. The ones which are shown in the snapshots were selected based on what was similar between the novice group and the expert. Segments which were discovered by the novice group at the end of the session are highlighted and the relations are marked by dashed lines in Figure 3.4. Segments which were elaborated by the expert are also highlighted against what was found by the novices. The additional relations are also marked by dash-dotted lines. Neither of the designers elaborated on a hierarchical structure of components except for the 'landing gear' which was decomposed into a 'beam' and 'wheels'. Therefore the group *Structure* was similar to the class component and for simplicity, it is left blank.

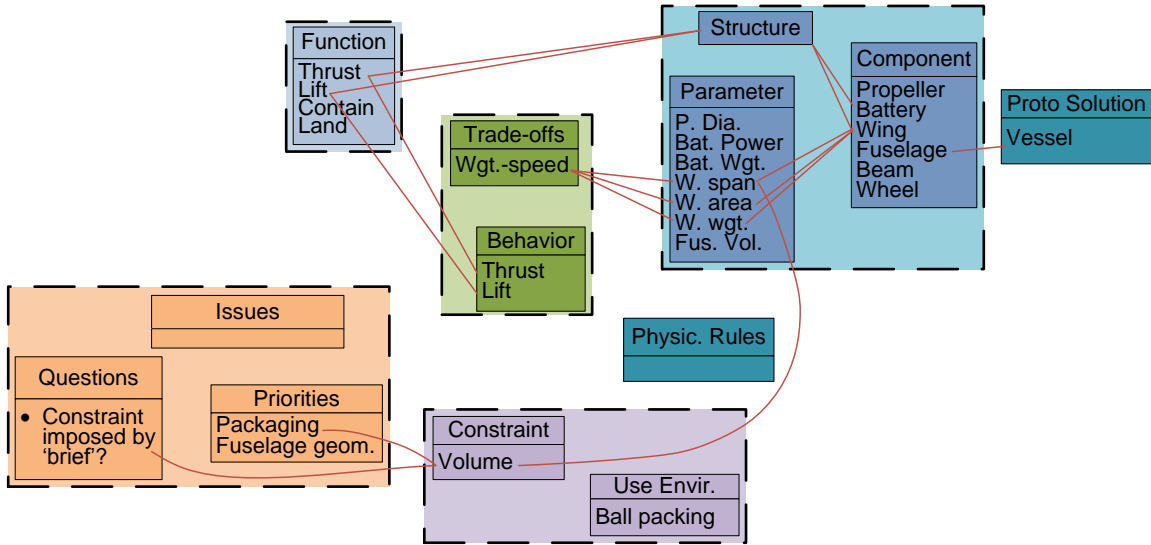


Figure 3.3 A snapshot of the novices halfway through their session (from [77])

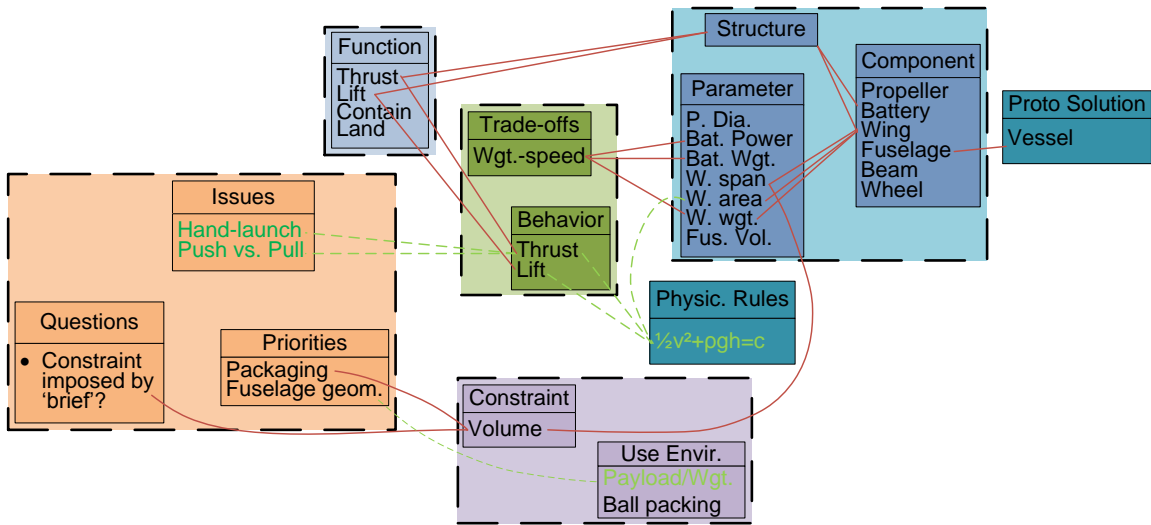


Figure 3.4 A snapshot of the novices at the end of their session (from [77])

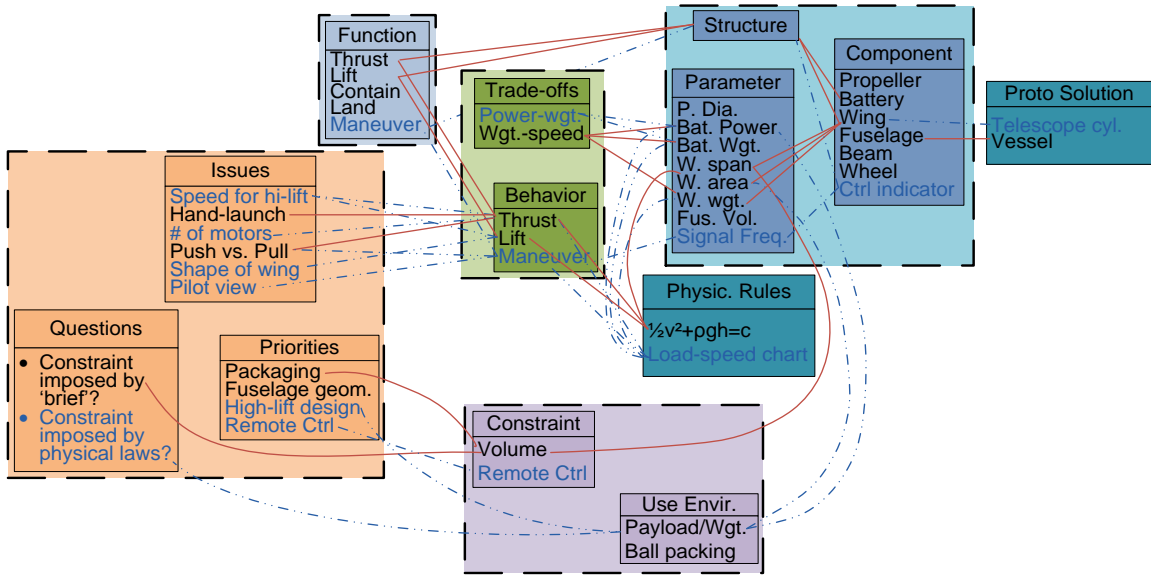


Figure 3.5 A snapshot of the expert at the end of his design session (from [77])

The initial ontology and structure of the representation were derived from the first exploratory study. The structured representation provided a way to capture and compare the problem formulation of an expert and a group of novices. The richness of the relations captured, by the expert, among different segments of different entities, in addition to some segments disconnected from other in the novices' formulation could be shown. There was still a need to see if this framework could be used to represent problem formulation for a different problem among new participants. Therefore, a second exploratory study was conducted.

3.2 Modified modeling structure

The objective of the second exploratory study was to check if the ontology and structured representation could be generalized to a different problem and designers, and what modifications were required. The initial framework had several groups associated with ontologies (mainly Function-Behavior-Structure) found in the literature [35, 78, 79].

The observations in the collected protocols did not exactly follow relations suggested in those ontologies. For example, Gero *et al.* [78] define behavior as a link between function and structure. But the coded segments assigned to parameters (under group *behavior*) belonged to components that were not always related to a function through a behavior. Thus, the grouping was abandoned in the modified model.

There were also changes in the new framework partly in accordance with new observations and partly for simplification. *System architecture* was removed; the hierarchy could be shown with explicit relations among components. *Tradeoff* and *Priority* were also removed. The former was merged with *issue*. The latter was a process entity that could be represented in changes through time. Understanding the importance of analogies in creative problem formulation [80], a new entity named *Analogy* was added. *Constraint* was substituted with a more general entity *Requirement*. Another modification to the model was to distinguish different types of relations among the entities. Four types were identified: covariation, when changes in a segment (within an entity or among entities) affects another segment; option, when new ideas emerge for similar concepts; instantiation, when a segment is added off of a previous segment; and substitution, when a new segment is added as a substitute for a previous segment.

The task for the second exploratory study was to design a mechanical device deployed from a row boat for taking water samples from a lake up to a depth of 500 meters. The data collection settings were similar to the first exploratory study. Data was analyzed for one expert and one novice student. Segmentation and coding was done similarly to the coding process described in the previous section. However, the coding schema was

predefined, i.e., a segment was given one of the entities or relations types which were already defined.

Table 3.2 shows a few examples of the coded segments corresponding to the predefined entities, taken from the expert's protocol. The first column shows the order of occurrence of each utterance. The second column is the corresponding entity. The third column is the extracted quote from the protocol. The next column is the label assigned to the segment which is used in a structured representation. The last column shows related observations. The total number of utterances for the design session was 108. The selected observations are taken from the first 13 minutes of the session which lasted 52.

Relations were also coded. An example of *covariation* was the relation between rupture pressure, depth, and the triggering function. In utterances 29 and 38, the wire was used in one design with a valve and in another design with a rod which showed *option*. A component in utterance 13 was immediately added as an *instantiate* of a proto-solution in the previous utterance. The proto-solution in utterance 7 was a *substitute* of the proto-solution in utterance 4.

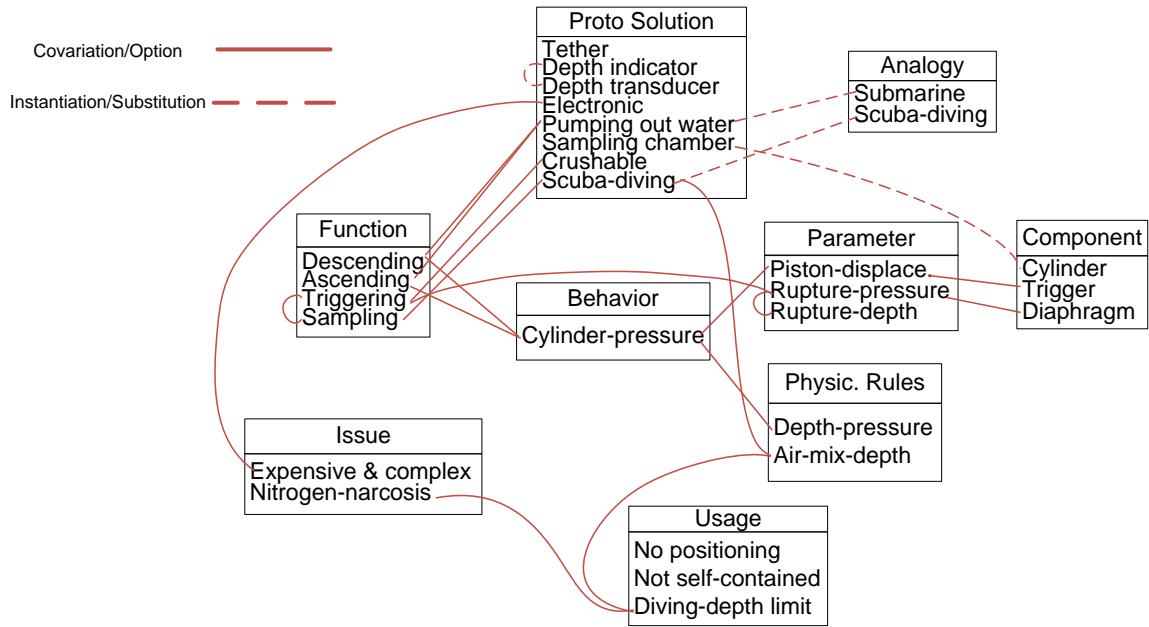
Exploring the literature to find appropriate representations for showing problem formulation data continued in this study with a focus on relations. There were a few inspiring representations: parent-child relations in ERD diagrams (e.g., between *Physical system architecture* and *Component*); class structure relations in UML models (e.g., between *Physical system architecture* and *Function*); or optional attribute relations in EXPRESS-G models (e.g., between *Use environment* and *Priority*). However the relations which were observed in the protocols would not entirely fit either of the mentioned representations. For example, in a class structure of a UML model, *Function*

would actually be a mechanism of a class of a *Physical system architecture* or *Component* object. Therefore, the final representation was similar to that of the first exploratory study where segments took a distinct label and put in a box under the corresponding entity.

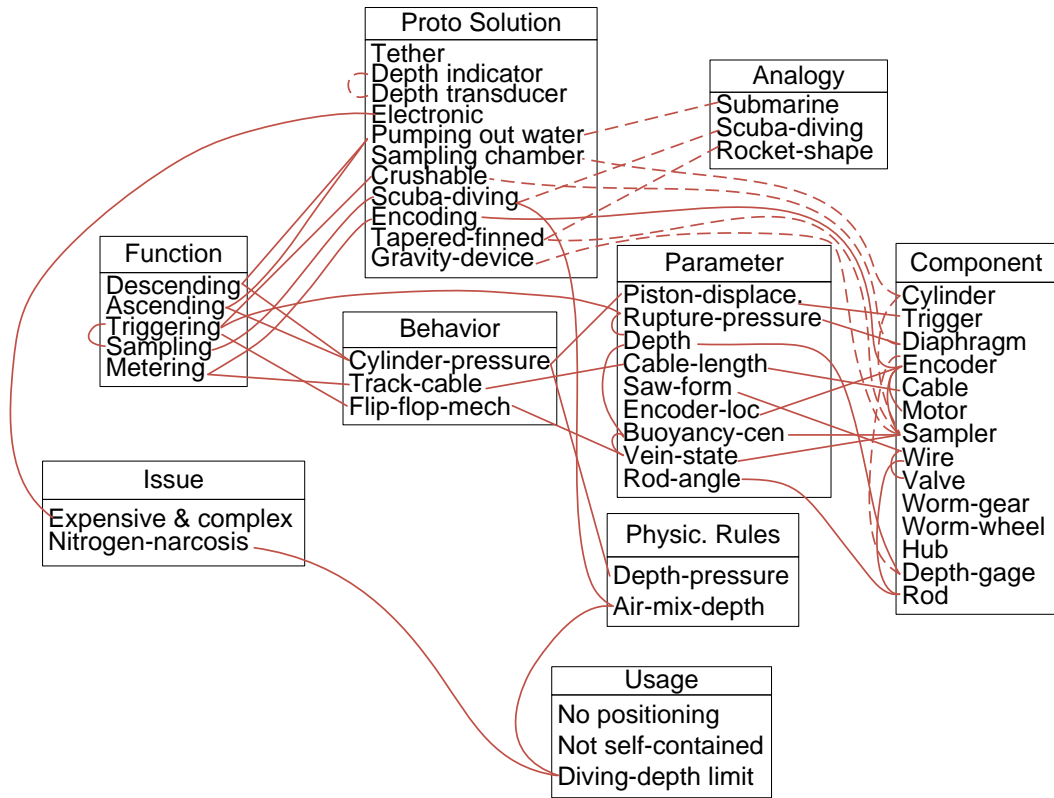
Similarly to the first study, representation models were drawn to show each designer's change in problem definition. Figure 3.6 and Figure 3.7 show two snapshots for each of the expert and novice problem formulations respectively (for a clearer example of showing evolution of a designer's problem formulation in time, see Appendix A). For each designer, about a quarter of their session is represented. When both designers refer to similar ideas in a segment, the same name is given for both designers to make comparison easier. There are a few entities that are not present in these snapshots for one designer (e.g., *Requirements* for the expert and *Physical rules* for the novice), or for both designers (e.g. neither of the designers have an example of the entity *Question*). The absence is due to the fact that the instances occurred in later stages of the design session. The expert designer revisited the requirements after about 19 minutes and raised a question after about 33 minutes.

Table 3.2 Coded segments from the first protocol analysis (from [81])

#	Entity	Observation	Label	Relate
2	Function	<i>it just needs to ascend and descend</i>	F: Descending F: Ascending	
4	Proto-sol.	<i>[concept] B is some sort of depth indicator</i>	PS: Depth indicator	
7	Proto-sol.	<i>So [concept] B is some sort of depth transducer</i>	PS: Depth transducer	4
12	Proto-sol.	<i>then the other main subsystem is the sampling chamber</i>	PS: Sampling chamber	
13	Comp.	<i>there is a hollow cylinder and one end is capped and there is a piston</i>	C: Cylinder	12
14	Behavior Parameter	<i>and this piston since this is filled with atmospheric air on the backside of the piston, atmosphere, atmospheric pressure you will pre-determine how far this piston travels which will determine the depth</i>	B: Cylinder-pressure P: Piston-displace.	2
15	Physic. rule	<i>we know that about 34 feet of freshwater is one atmosphere so you can determine how many atmospheres of compression that you want the system to move before you trigger</i>	Ph: Depth-pressure	14
17	Function	<i>it tells the sampling to go ahead and take the sampling</i>	F: Triggering F: Sampling	16
19	Comp Function Parameter	<i>it has a diaphragm that ruptures at a specific pressure</i>	C: Diaphragm P: Rupture-pressure	15,17,18
20	Parameter	<i>that diaphragm ruptures when it gets down to a pre-determined depth</i>	P: Depth	19
24	Physic. rule Usage Issue	<i>500 meters is about 1500 feet so that is well beyond the limits of normal air and nitrogen mixture they will have nitrogen narcosis</i>	Ph: Air-mix-depth U: Diving-depth I: Nitrogen-narc.	23
26	Proto-sol.	<i>So the tethering can obviously be, it can be electronic</i>		3,8
27	Comp Function	<i>it could have some sort of encoder that meters out the cable and some sort of motor</i>	C: Encoder C: Cable C: Motor F: Metering	26
28	Behavior Parameter	<i>keeps track of the amount of cable that is extended</i>	B: Track-cable P: Cable-length	27
29	Function Comp Parameter	<i>And then when the sampling device gets to a certain depth it can have a wire that is wound into the cable that opens the sampling and can have a saw that opens the sampling uh, sampling valve</i>	C: Sampler C: Wire C: Valve P: Saw-form	17
38	Function Behavior Parameter	<i>So when the device is going down the last 10 feet of the cable is a steel rod, perhaps and then it transitions, the cable transitions into the rod for some period of distance. And then the sampling device uses the angle between the rod and the sampling device to open a valve</i>	C: Rod P: Rod-angle	27,29,37

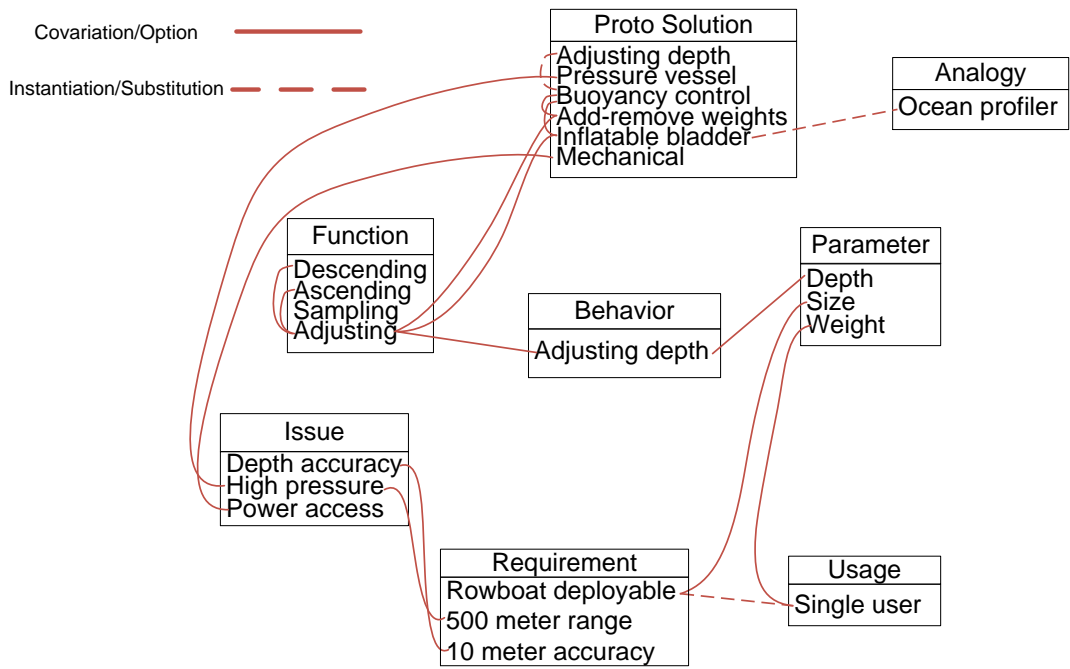


(a)

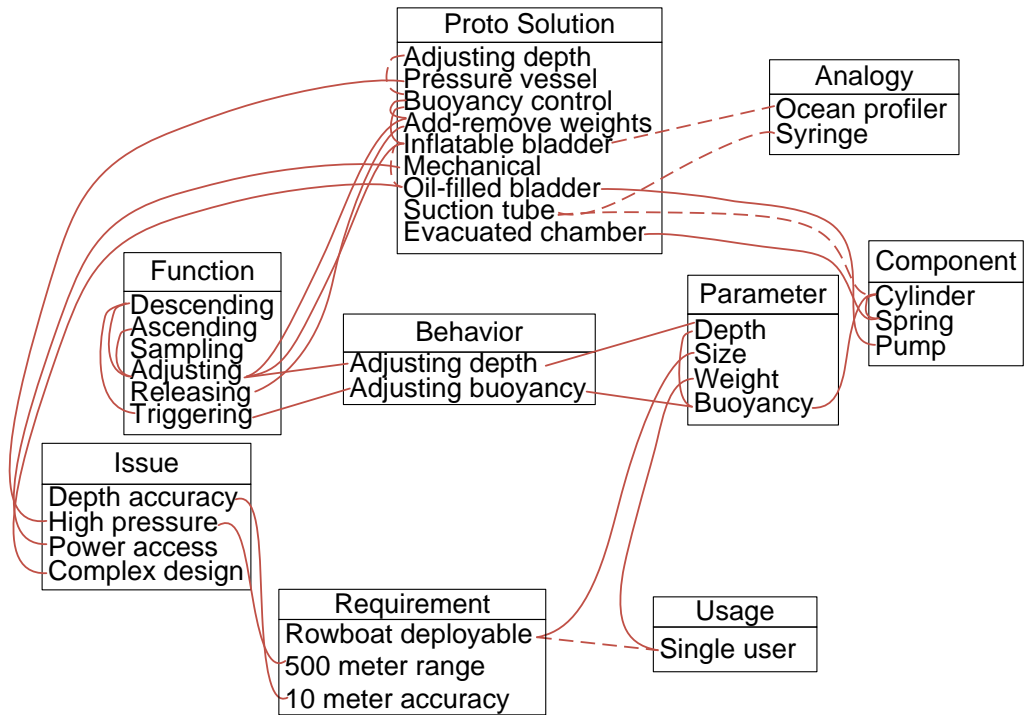


(b)

Figure 3.6 Expert's formulation after 8 (a) and 13 (b) minutes (from [81])



(a)



(b)

Figure 3.7 Novice's formulation after 11 (a) and 17 (b) minutes (from [81])

The second exploratory study showed that it was possible to use the structure created in the first exploratory study (with slight modifications) for a different problem and participants. The representation demonstrated the richness of the expert's problem formulation compared to that of the novice, in terms of more expressed entities and relations. The two modeling framework from the exploratory studies were based on data collected for two problems and a few designers. Though the relatively similar structures showed potential for generalizing the framework to other problems, there was still a need for studying other possible entities that could go into a representation of problem definition, missing from the two specific examples in the exploratory studies.

3.3 Synthesizing the models from the exploratory studies

The final step towards creating a framework for representing designers' problem formulation was to synthesize the entities, relations, and representation structures which were explored in the previous steps. To ensure that the search for relevant elements of the framework was not limited to the observations of the two protocol studies, additional entities were added through brainstorming. A few researchers in the brainstorming sessions brought in their experience from years of studying or teaching engineering design. I had specifically studied several design textbooks such as Ulrich and Eppinger [82], Dieter and Schmidt [83], Pahl and Beitz [31], and Norman [84]. Entity names were written on sticky notes and put on a wall, see Figure 3.8.

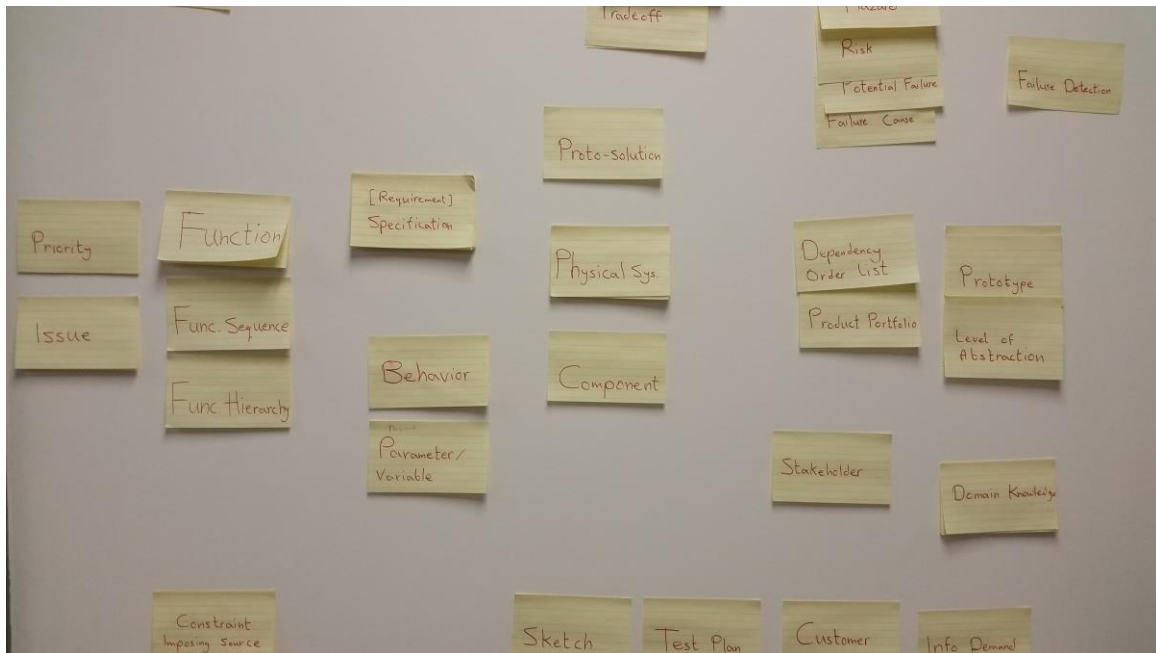


Figure 3.8 A collection of entities from brainstorming

The collection of entities was synthesized using the affinity method, i.e., by merging similar entities iteratively to reach consolidated groups. Definitions of entities were discussed to merge close entities. An example is merging the following entities into requirement: requirement, specification, goal, constraint, objective, [customer] need, wish, and demand. Each of these entities has slightly different definitions. However, there is not a single definition for each entity and different textbooks might use common terms to refer to slightly different things or vice versa. Ulrich and Eppinger [82] define *need* as what the customer wants independent of any particular product that will be designer, while *specifications* depend on the selected concept. They state that they do not make a distinction between want and need. They also mention that *attributes* and *requirements* are also common terms used in practice to refer to *need*. Pahl and Beitz [31] divide *requirements* into *demands* and *wishes*. Demands are requirements that if not fulfilled

render the design unacceptable. Wishes are requirements that should be considered when possible. They also state that a requirement can be either quantitative or qualitative while Ulrich and Eppinger [82] attribute quantification only to specifications. There is a tradeoff between having more entities to express things more distinctly, and making it easier to learn, remember, and use them in expressing things. It is also possible to use common entities for close things but assign attributes such as subtype to have the desired distinction. Through multiple group discussions, the entities were narrowed down; see Figure 3.9.

In addition to merging similar entities, a common structure for relations was also laid out. In the final analysis, the similar relations were those of inter-entity or intra-entity. Inter-entity relations can be considered as parent-child or hierarchical relations. Intra-entity relations can be considered links or have particular names. Reaching this common structure was not entirely driven by trying to merge similar entities or relations. During the development of the framework, some specifications of a desired modeling framework for representing differences in designers' problem formulation were derived. They are discussed in the next section.

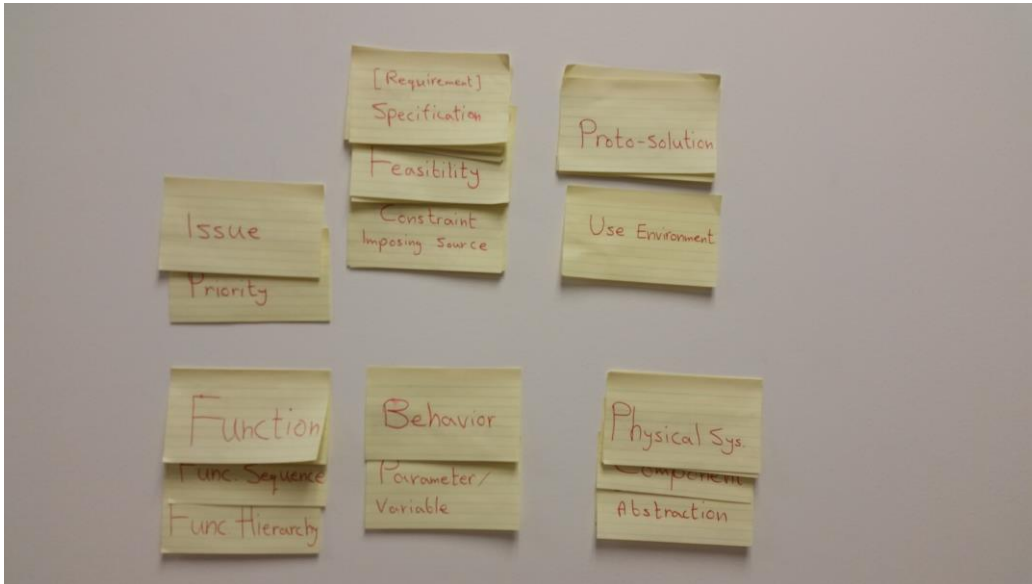
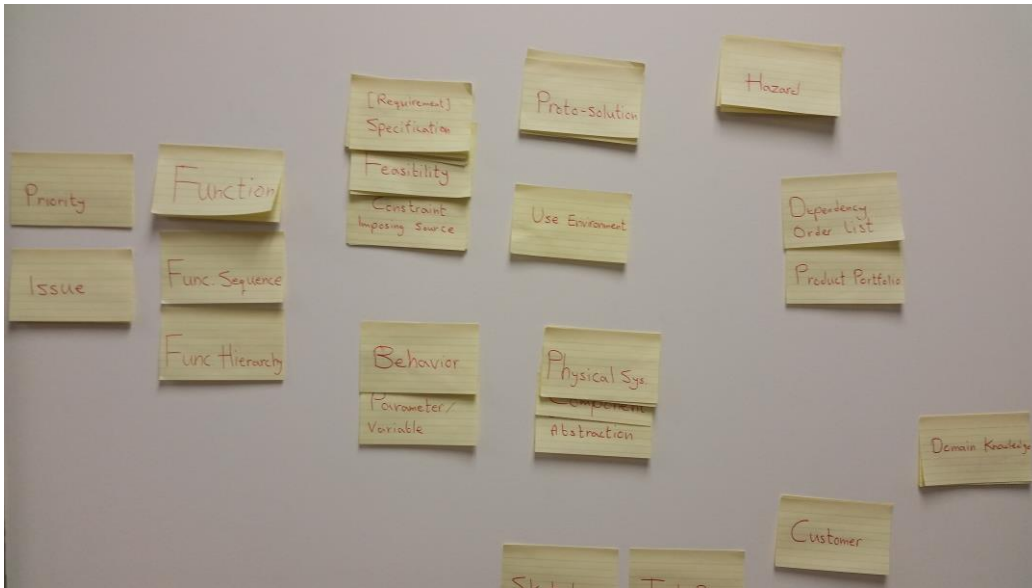


Figure 3.9 Merging entities in multiple steps

3.4 Specifications of the modeling framework

The objectives which were set out at the beginning of the process of creating a modeling framework for problem formulation were at a high level. Basically there was a need for a structure which facilitated showing differences in how designers formulate problems. Discovering the specifications of the tentative framework was part of the

process which I discussed in previous sections, i.e., exploration, refining, and synthesizing entities. These specifications are shown in Table 3.3. They are described here:

- 1- The problem and solution spaces co-evolve in design [4], therefore, the framework should be able to model artifacts, and behaviors (solution-oriented concepts in designing), in addition to requirements and functions (problem-oriented concepts).
- 2- Another desired feature is representing hierarchal structure, since designers divide problems and solutions into sub-problems, and sub-solutions, in order to cope with complexity and evolution (change in sub-systems) in design [85].
- 3- Designers can divide problems in multiple alternatives ways, and combine different sub-problem and sub-solutions, thus the framework should allow multiple and disjunctive compositions.
- 4- In relation to compositions, one also should be able to model sequences within the framework. In functional decomposition, different choices of sequences of common functions lead to different designs. For example, in designing an automatic brake, the sequence of the sensing and the braking functions have significant consequences to the safety of the brake.
- 5- Designers link different fragments during problem formulation. Identifying links among design entities relates to creativity [46, 86]. Therefore, the framework should enable linking entities of different types.
- 6- The framework should be domain independent. The scope of the examples in this research is mostly the design of mechanical products with focus on the

conceptual design of new products, and not variants. However, most product designs have electrical and electronic elements and it is difficult to separate domains in an actual design process. It should be possible within the framework to express problem formulation of a combustion engine with its known behaviors, or an engine with an abstract function of providing power, including but not limited to a solar-powered motor.

Table 3.3. Specifications of a framework for problem formulation

Specifications	Justification
Problem and solution oriented	Co-evolution of problem and solution spaces during problem formulation
Hierarchal	Describing compositions and levels of abstraction
Disjunctive	Considering alternatives with common or independent fragments
Sequential	Showing precedence in one level of abstraction
Linked	Showing relations among different types of entities
Domain-independent	Describing problems with generalized categories common to different engineering domains

Besides the listed specifications, the tentative framework can be examined with respect to a few measures of goodness. If the specifications are characteristics which the framework should have, measures of goodness are characteristics which the framework should be better at compared to other frameworks. Three measures are proposed with potential methods of evaluating them (see Table 3.4): expressiveness evaluated with coverage; compactness evaluated with entropy; unambiguousness evaluated with inter-rater agreement.

- 1- Expressiveness: In order to show how differently designers of different levels of creativity and experience formulate problems, the framework should represent enough level of detail to enable such comparisons. The resulting data models should be easily created and translated, but should also not lose valuable information that uncovers patterns of successful or weak formulations.
- 2- Compactness: This is a relative measure but it provides hints to including some entities with similar properties in the same group or class. For example, one may consider safety and ergonomics as different sources of defining requirements. A long-term objective of this research is to analyze data, collected from a large number of participants. A more compact yet fine-grained model makes automated analysis with computers faster, as well as easily exchangeable among different software tools. In addition, it is easier for a designer to learn the elements of a more compact data model in order to categories one's thoughts with respect to the framework.
- 3- Unambiguousness: This has two implications. First, if the framework is used as a coding schema to represent protocol data, different coders should have a close agreement in coding the same fragments (inter-coder reliability). Second, if a user is directly asked to categorize his or her thoughts within the framework, the chosen categories should not be very different from what a coder would interpret of those thoughts.

To a degree, these measures are related to each other. There is a balance between expressiveness and unambiguity based on the level of granularity determined in the framework which affects compactness. Unfortunately, except for inter-rater agreement,

the proposed evaluation methods are not common in design research. For this reason, only measuring inter-rater agreement was pursued in evaluating the proposed framework. This will be discussed in the next chapter with the introduction of the Problem-Map framework.

Table 3.4 Measures of goodness for the tentative framework

Measure of goodness	Potential evaluation method
Expressiveness	Coverage of coded fragments (ratio of coded to total)
Compactness	Information content (entropy) of coded fragments
Unambiguousness	Inter-rater agreement

CHAPTER 4

THE PROBLEM MAP ONTOLOGICAL FRAMEWORK

The previous chapter discussed the process which led to the creation of a modeling framework for representing problem formulation. This chapter describes the result: the Problem-Map (P-maps) ontological framework. The data model of the framework is described in addition to some changes to improve it based on initial applications. P-maps are compared to a few pertinent modelling frameworks with respect to the specifications described in the previous chapter. This is to validate the need for a new framework. Once P-maps are validated, they can be used in the experimental studies to test the research hypotheses. The exploratory models described in the previous chapter could not be used for that purpose.

4.1 Initial data model

The data model of the P-maps framework has evolved through the process which was described in the previous chapter. It started from a simple set of entities to a few groups of entities, attributes for each entity, and with a common structure including hierarchical within-group relations and inter-group relations. Each group consisted of entities whose instances could also be a part of disjunctive hierarchies.

The initial model incorporated five types of entities: Requirement, Function, Artifact, Behavior, and Issue. All groups were inter-related with bidirectional relations. Figure 4.1 shows the structure for this version of the P-maps data model. For the sake of simplification, only one direction is shown (which can be read as an active verb) in each

of the relations, e.g., it is shown that an artifact “realizes” a function while the relation that states a function “is realized by” an artifact is not shown.

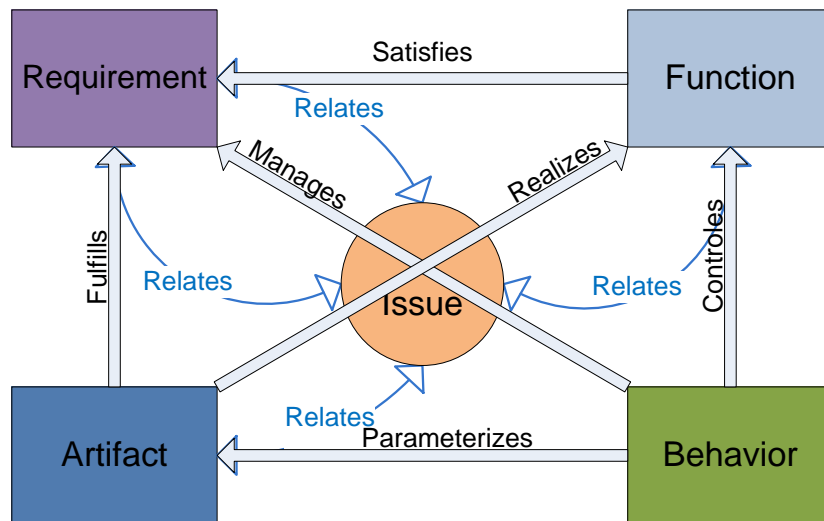


Figure 4.1 The first structured P-maps framework

This version of P-maps served the main objective of the research, i.e., showing differences in designers’ problem formulation. Some of the experimental studies which will be described in Chapter 7 and Chapter 8 use this version. However, the model was too abstract and a finer level of granularity was needed. This was mainly because of not fully exploiting attributes as intended. Attributes were supposed to provide more details to entities. Through the refinement and synthesis process described in Chapter 3, it was suggested that similar entities that were combined into one entity could be distinguished by a ‘subtype’ attribute, therefore details could still be added. A specification with a certain level (e.g., payload of 1000 pounds) could be defined as a requirement with a ‘subtype’ attribute named specification and a ‘level’ attribute with a value of 1000 pounds. The problem of not using attributes was more apparent during experimental studies when designers were asked to express problem formulation as fragments within

the ontology compared to when designers freely expressed their thoughts and researchers coded them. Therefore, the model was updated.

4.2 Improved data model

To address the problems described above, the first version of P-maps was updated with adding a new group of entity, Use scenarios, more details about entity subtypes, and by specifying a few attributes for each entity. The need for including Use scenarios in the data model was based on the importance of two notions in the conceptual design process: situatedness [87], or how the environment affects the design; affordances [84], or how users interact with the design.

Some of the changes were motivated by the methods found in design textbooks. Ulrich and Eppinger state that in order to identify customer needs, the designer should experience the use environment of the product [82] which is another reason for adding use scenarios. Some of these methods propose a formal output. Examples are objective trees, spec sheets [31, 82], and function trees [32] which were described in section 2.1. Figure 4.2 depicts the updated data model for the P-maps ontological framework. The attributes shown for each entity are examples and are optional. Other attributes can be added if necessary. The names given to the relations between any pair of entities might be overwhelming; with the addition of the Use scenarios, the number of inter-group relations increased from 7 to 11 relations. One option is to name the relation by combining names of the pairs, e.g., requirement-function. The definitions of the entity subtypes are described below. For each entity, there is a subtype with the same name as the entity.

This is to include a general definition of that entity for cases other than the other defined subtypes.

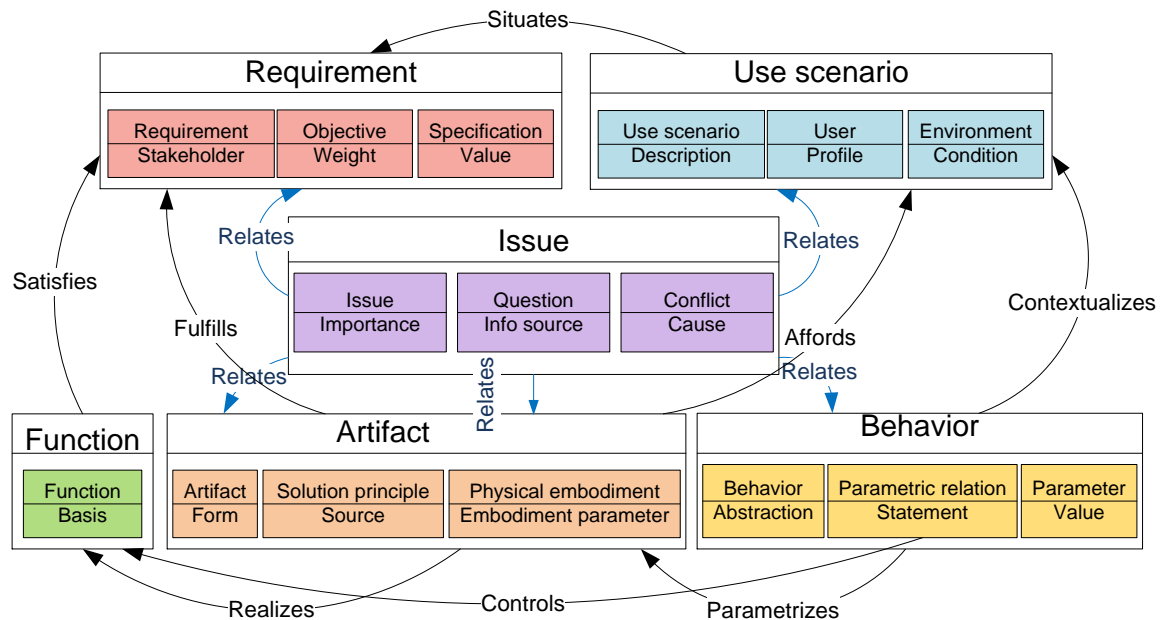


Figure 4.2 The data model for the updated P-maps ontology

As discussed in section 3.3, there are no standard terms used in textbooks for these entities. Although some terms such as objective, spec, or requirement are used interchangeably in the literature, particular definitions are given to the entities in this research. In P-maps, requirements are the entities that describe what the design should achieve. A design problem is usually given as a design brief or problem statement. The design problem is formulated with additional requirements elicited by the designer. A naming convention to follow is to start the phrase with imperative modal verbs such as “should”, “must”, or “has to”. There are a variety of terminologies in defining requirements including objectives, targets, constraints, specifications, and requirements. In P-maps one of the following categories should be chosen:

- Objectives [**“obj”**]²: a measure of goodness or fit that can be used as a criterion for assessing different designs in comparison to each other. Preferred designs are ones that have better outcomes with respect to the objectives of higher importance. An example of an objective is a lower cost or longer life-cycle. The objectives can be structured in a hierarchy which enables the designer to create an objective tree in the web tool.
- Specifications [**“spec”**]: design attributes that specify a level of desired performance. An example of a spec is a payload of 5000 kg, or a speed of 100 miles per hour. In P-maps there is no separate subtype for constraints and the designer may specify constraints with specific target ranges or double bounds as specs (e.g., available power 10 ± 2 KW, available gear ratios 3:1, 9:2, and 5:1, or use 110V AC) and constraints with single bounds as objectives (e.g., $\text{cost} < \$100$). It is also possible to simply not choose any subtype in which case the general category of requirements is selected.
- Requirements [**“req”**]: any other desired attributes e.g., legal requirements, material requirements, etc. or what the designer cannot put under objectives or specs as defined above can go under requirements.

² The highlighted abbreviations are optional tags which help to specify the subtypes in each entity group. Users of the web-based tool associated with the ontology are encouraged to use these tags and the described naming conventions to make evaluation of the data (either by a human judge or an automated text processing program) easier.

Use scenarios describe how and where the design is used; in other words, the users who will be using the design outcome and where and under what conditions the design will be used. More specific subtypes can be expressed as:

- Users [**“user”**]: who is the design for. This may include demographics of the target users such as age and gender; whether they have special needs because of disability; how users interact with the design and the human sensory receptors involved.
- Environment [**“env”**]: where and under what condition is the design used. This may include geographic information about the target customer base, e.g., the level of humidity that may cause rusting; other environment variables, e.g., the change in temperature at the altitudes where an airplane cruises.
- Use scenario [**“use”**]: any general description of a possible scenario of using the design.

Functions refer to what the design does and the actions that the design will execute. A naming convention to follow is to use verbs or verb noun combinations e.g., sink, move, rupture disk, carry passenger, or amplify torque. The hierarchy represents functional decompositions [88]. P-maps incorporate disjunctive composition, making it possible to have multiple functional decompositions using common sub-functions. There are no subtypes under this category.

Artifacts describe what is created or used to realize the functions of the design. They include the physical embodiments (physical systems, parts, or of-the-shelf components) or the solution principles that the design may be using. P-maps allow compositions of

solution principles and physical embodiments. The hierarchy resembles a product architecture.

- Solution principle ["**sl**"]: analogies, e.g. submarine for a sinking object with reservoir; principles such as lever, or nesting tubes (one of the 40 TRIZ principles [89]).
- Physical embodiment ["**em**"]: off-the-shelf part or component e.g. gear, motor.
- Artifacts ["**art**"]: any general description of a system that is used to achieve the design requirements, use scenarios, or realize the functions.

Behaviors are the physical properties and laws that govern the design. These entities include parameters (design variables), and parametric relations (known relations among variables expressed in mathematical equations, or qualitative relations that the designer knows among parameters). Behaviors may be expressed by parametric relations, which are composed of sets of parameters.

- Parameter ["**par**"]: design variables, e.g., pressure, speed, motor rpm.
- Parametric relation ["**eq**"]: known mathematical expressions e.g., $\Sigma = F/A$, or Newton's first law; relations expressed qualitatively, e.g., hydrostatic pressure is related to depth.
- Behavior ["**beh**"]: any general description that is not defined using a parametric relation.

An issue is a point that the designer believes to be pivotal or problematic in achieving a design objective. An issue can arise in realizing a function with a specific artifact or behavior, in realizing conflicting design goals such as lower weight and strength of a

structure or in accommodating different components in a product architecture due to incompatible interfaces to name a few. The designer gains insight in the discovery of key issues in the design and the areas of the design that should be prioritized.

- Conflict ["**conf**"]: e.g. a conflict between minimizing weight and maximizing strength.
- Question ["**q**"]: feasibility questions from self or an expert e.g., is a flying device an option given the limited power sources; missing information questions from client or user e.g., what surface will the device move on or how many motions can a user simultaneously control with two hands.
- Issue ["**iss**"]: any other issues.

Hierarchies and partial orderings manifest intra-group relations. Inter-group relations are also defined; an underlying property of ontology. This leads to a model that can show how different designers see the relations among different aspects of a problem and the alternative ways they relate. For example, alternative conceptual designs with common components or different function decompositions can be shown with different branches of an artifact or function hierarchy with nodes that have the same name for the common components or functions respectively. A specific name is assigned to the relation between any of the two entity groups. For example, an artifact realizes a function, and a behavior manages a requirement. The P-maps model does not make a distinction between explicitly known relations (e.g. when a designer knows that the power equation of an electric motor manages the desired torque), and implicit or qualitative guesses (e.g., when a designer knows that a parameter manages a specific goal but does not yet know how exactly). Having hierarchical and linked structures were two of the specifications desired

for a framework developed for this research. P-maps can be validated with respect to those specifications in comparison to other modeling frameworks.

4.3 Model validation

In research, validity generally refers to whether the methods used or conclusions drawn in a study are relatively accurate and correspond to the subject phenomena [90]. The motivation behind creating the P-maps framework was that existing frameworks could not serve the objectives of this research. To validate P-maps, it should be demonstrated that other frameworks cannot provide an enough accurate representation of problem formulation. Specifications of a framework for studying problem formulation were discussed in section 3.4. In this section, a few frameworks which were reviewed in Chapter 2 are compared to P-maps with respect to the aforementioned specifications.

The specifications were that the framework should accommodate problem and solutions elements, hierarchical structure with sequences and disjunctive branches in addition to links between different types of entities. Table 4.1 compares P-maps to relevant modeling frameworks with respect to the specs. These frameworks were chosen because they have been highly used in research in conceptual design (F-B-S and Linkographs), or in representing problems (Four-box and SysML). Concept Maps have all the desired representation characteristics but are for general purpose knowledge representation. All frameworks can model links among entities, thus this spec is omitted.

In section 3.4 three measures of goodness were also specified: expressiveness, compactness, and unambiguity. In order to objectively compare P-maps to other relevant representation frameworks there are two possibilities. One is to give a piece of protocol

to an independent coder (or coders), i.e., a coder who has not participated in developing neither P-maps nor the other ontologies, teach them the two coding schemas to be compared, and ask them to code the protocol. The other is to ask two researchers who have contributed to the two ontologies to work together and code a protocol in each ontology. The coded protocols can then be evaluated with respect to the measures proposed in Table 3.4. It was not possible to conduct a comprehensive comparison with an independent coder or with coders involved in the other ontologies, though this can be a part of future work. Yet, examining how a protocol already coded in another ontology can be coded in P-maps is useful to show its representation power.

Table 4.1. Comparison of different modeling frameworks to P-maps

Spec Framework	Problem and solution oriented	Hierarchal	Disjunctive	Sequential
F-B-S	Problem & solution	No	No	No
Four-box	Problem & solution	No	Implicit	No
Linkograph	Solution-oriented	No	Explicit	Yes
SysML	Problem & solution	Yes	Implicit	Yes
Concept map	n/a	Yes	Explicit	Yes
P-maps	Problem & solution	Yes	Explicit	Yes

Consider the piece of protocol transcript in Table 4.2 which is coded within Function-Behavior-Structure and P-maps (the comparison is taken from [2]). The protocol and its F-B-S encoding was done by Gero and Mc Neill [35]. In the second column, apart from coding segments as F, B or S, the requirements are also coded as R and the level of abstraction is also identified (0 - System, 1 - Input Block, 2 - PAL Block, 3 - Output Block). The fragments encoded within P-maps are shown as Prolog logic predicates as the

formalism provides simple readability (it will be shown that the formalism is used in formalizing and tracing strategies; here, they are chosen out of convenience, otherwise they could have the coded segments could be shown differently e.g., for the first segment ‘solution principle: input_block’). The protocol is coded into 6 fragments within F-B-S but there are 22 P-maps fragments where:

- 6 fragments represent inter-entity links (encodings with the heads: connects, realizes, fulfills, and manages).
- 2 fragments represent a 2 level deep hierarchy (output_block is the parent of dalington_driver which is the parent of optical_dalington).
- 2 fragments represent 2 disjunctive branches (input_block is the parent of opto_couplers in one segment and the parent of external_pull_ups in another segment).
- 1 fragment represents an attribute specifying another fragment (goal target for number of outputs is 8, i.e., number of outputs should be more than 8).

Table 4.2 A protocol coded in F-B-S [35] compared to P-maps (from [2])

Fragment	F-B-S	P-maps
what we need is some sort of input block there. The PAL, there might be one or two other bits around it, I don't know, and the output block. And that's the fundamental picture of what we're going to have to do.	0S	solution_principle(input_block) physical_embodiment(PAL) solution_principle(output_block) connects(input_block,PAL) connects(PAL,input_block) parameter(number_of_bits_around_PAL)
darlington driver if at all possible, an optical darlington driver,	3S	parent_of(output_block,dalington_driver) parent_of(dalington_driver,optical_dalington)
The input block is...really fairly straight forward...opto couplers	1S	parent_of(input_block,opto_couplers)
With of course external pull-ups I guess so that we can operate on any voltage.	R1S	solution_principle(external_pull_ups) parent_of(input_block,external_pull_ups) function(operate_on_voltage) realizes(input_block,operate_on_voltage) requirement(flexible_input_voltage) fulfills(external_pull_ups, flexible_input_voltage)
That's one of the ideas of putting that input block onto"?"? not only the safety side but the flexibility side as well. That's the other reason of course for opticals on that side.	R1F	parameter(location_of_input_block) requirement(safety) manages(location_of_input_block,safety) manages(location_of_input_block,flexible_in put_voltage)
My minimum requirement would be for 8 inputs minimum...8 inputs sorry 8 outputs minimum	R2S	goal(number_of_outputs) goal_target(number_of_outputs,more_than,8)

To know the degree to which human interpretation affects understanding of a coding schema, it is common to find the inter-rater agreement, i.e., to examine how different raters agree on coding a corpus with regard to the ontology. In this context, the ontology is P-maps (as a coding schema) and the corpus is coded protocols. Two statistical measures of agreement in assigning categorical ratings are Cohen's kappa [91] and Fleiss' kappa [92]. Both measures take into account agreement occurring by chance. They range from zero to one, zero representing no agreement, one representing perfect agreement. Cohen's kappa is used for two raters while Fleiss's kappa is for any fixed number of raters.

To measure the inter-rater agreement in coding protocols with P-maps, segments of code were given to trained raters. Raters assigned each segment to one of the categories {requirement, function, artifact, behavior, issue, hierarchy, inter-group} in P-maps. The equation for Cohen's kappa is:

$$\kappa = \frac{P(a) - P(e)}{1 - P(e)}$$

where $P(a)$ is the relative observed agreement and $P(e)$ is the probability of agreement by chance. Fleiss's kappa is computed similarly and $P(a)$ and $P(e)$ are found from:

$$P(a) = \frac{1}{Nn(n-1)} \sum_{i=1}^N \sum_{j=1}^k n_{ij}^2 - Nn$$

$$P(e) = \sum_{j=1}^k p_j^2$$

$$p_j = \frac{1}{Nn} \sum_{i=1}^N n_{ij}$$

where N is the number of coded segments, n is the number of raters, k is the number of categories, and n_{ij} is the number of the raters who assigned the i -th segment to the j -th category.

To determine the inter-rater agreement 6 segments were chosen from protocols collected from eight designers working on one problem (total of 48). Number of segments was based on Gwet [93] for an expected agreement of 70% among the coders, and an expected 20% error in coding for each rater. Six segments were found using systematic sampling. The total number of segments in each of the eight protocols was divided by six to find the interval for systematic sampling. Then the first segment would be at a random location between the start of the protocol and the length of the interval. The other five segments were found by adding the length of the intervals to the starting segment.

Three coders were familiarized with the ontology. Fleiss's Kappa for the three raters was 0.35, which is fair-moderate agreement [94]. A pairwise comparison with Cohen's Kappa, resulted in 0.41, 0.36, and 0.28 agreements between the pairs. Coding the relations was inherently more difficult because relations were vaguer to describe verbally and often related to entities which happened distant to each other temporally. After removing {hierarchy, inter-group} from the choices, the agreement would become higher: Fleiss's Kappa 0.48 for the 3 raters; Cohen's Kappa, 0.56, 0.47, and 0.43.

In addition to these three coders, inter-rater agreement was also measured between two of the researchers who were intimately involved in this study (throughout the development and application of the ontology). They were more familiar with the coding schema and not surprisingly, inter-rater agreement between them was higher. Cohen's

kappa for these two researchers, including the hierarchies and inter-group relations, was 0.64 which is substantial [95]. Excluding the relationships, the agreement would be 0.75.

This chapter described the detailed data model of the P-maps ontological framework. The model was compared to a few pertinent frameworks with respect to previously identified specifications. A thorough and unbiased comparison requires additional work in collaboration with researchers or communities who have created or contributed to those frameworks. However, it is still possible to show through examples how P-maps are more expressive in capturing different types of relations, specifying attributes for entities, and representing alternatives. Inter-rater agreement was found by asking raters to segment and code protocols collected from a few designers. Instead of asking raters to code a designer's thoughts on formulating a problem, an alternative way is to ask designers to code their thoughts within the P-maps ontology. Designers can express their thoughts within P-maps on paper. However, the process can be improved by using a computer tool, given the structured data model and representation of P-maps. This tool was created for P-maps. It is presented in the next chapter.

CHAPTER 5

THE PROBLEM FORMULATOR TESTBED

The main objective of this research is to learn what designers think about when they formulate problems. Review of the literature showed how few dedicated studies of problem formulation were. A survey of studies of designer thinking also showed that in general, empirical studies are based on few observation with few participants [2]. This is because a majority of empirical studies of designer thinking use the protocol analysis method which is resource-intensive. To collect and analyze data on a large scale in a shorter amount of time an alternative data collection method was needed. Since P-maps benefit from a structured data model and representation, using a computerized data collection testbed was feasible and promising. This chapter briefly explains the process of creating the Problem Formulator testbed and its features.³

5.1 System architecture

The Problem Formulator testbed is the means for collecting problem formulation data structured within the P-maps ontological framework. The testbed supports designers in constructing problem formulations with its interactive design assistant and additional features, e.g., generating reports. Nevertheless, the main purpose it serves is in speeding up data collection and analysis. The tool focuses on the early stages of the design process and lets the designer easily input information about their conceptual designs, store this

³ Though I have contributed to the design of the database and user interface, the implementation was done by Glen Hunt, Chris Maclellan, and Pradeep Mani.

content for later review, and display it for the user's inspection. Based on the modeling specification described earlier, a number of components were considered for the tool:

- An internal representation for encoding problem formulations.
- A graphical or textual notation for displaying a given problem formulation to the user.
- Operations for creating problem formulation entities.
- Operations for creating hierarchies within a type of entity.
- Operations for linking pairs of entities.
- Operations for editing and deleting entities and links.

The Problem Formulator was implemented in a manner that lets users access it from the World Wide Web. There were two reasons for this choice. First, making the software available on the Web makes it more accessible; users can run it from any location and on any device that operates with a modern Web browser (no additional software needs to be installed on the user's computer). Second, all entities and links that the user enters are stored in the cloud, where they are backed up and easily retrieved for future use, regardless of location or device.

To provide Problem Formulator with these features, CakePHP (a model-view-controller framework) was utilized along with a combination of PHP, JavaScript, MYSQL, HTML, and CSS. CakePHP was chosen because the MVC framework makes the software more modular and easier to develop and maintain. Figure 5.1 shows the four components that make up Problem Formulator: the stored problem formulation, the controller, the view, and the inference backend. The system encodes problem formulations internally in the problem map ontology, which consists of different entities

and relations among them. Problem Formulator stores this content in the relational database structure shown in Figure 5.2.

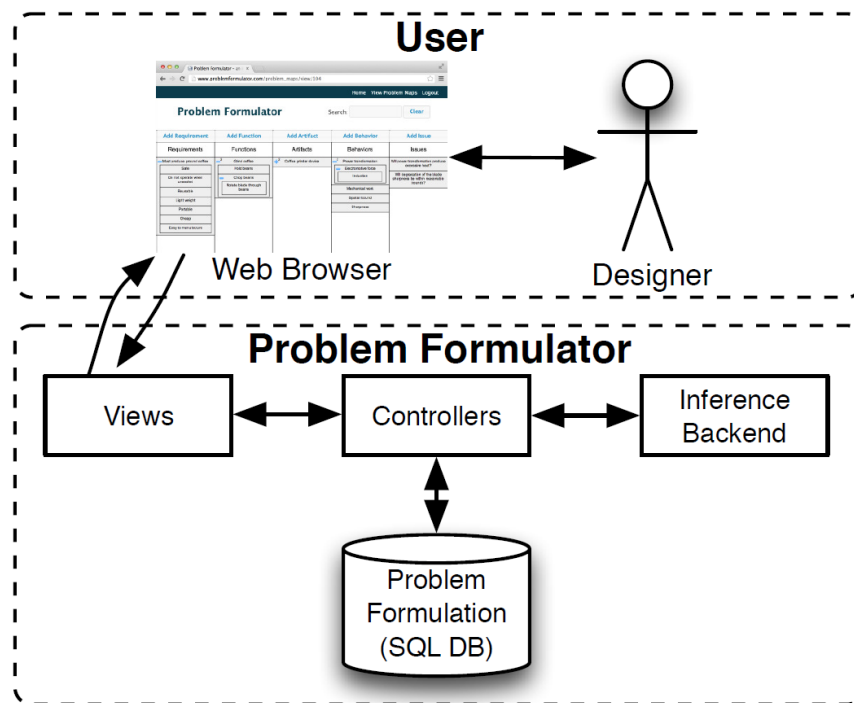


Figure 5.1 The system architecture of the Problem Formulator (from [96])

The view determines how Problem Formulator displays information stored in the problem map to the user. There are views for all basic functions, such as adding and deleting entities and links, as well as ones for the user’s active projects. The controller determines what information from the problem map is available in each view. There are controllers for creating and manipulating problem formulations, entities, and links; these provide a layer of abstraction between the problem map model and the view that ensures data integrity. Finally, the inference backend incorporates reasoning methods that lets Problem Formulator trace certain formulation characteristics (more specifically, these are formulation strategies which will be described in section 6.3). The designer connects to

the Problem Formulator through a Web browser, which displays their problem maps. When a user takes action in their browser, the changes to the problem map are sent to the server where they are stored in the database. Meanwhile, the interface is dynamically updated using Javascript, so the user never has to refresh their browser. If the Problem Formulator generates any feedback for the user, then it is sent to the web browser, which dynamically updates the problem map with the feedback.

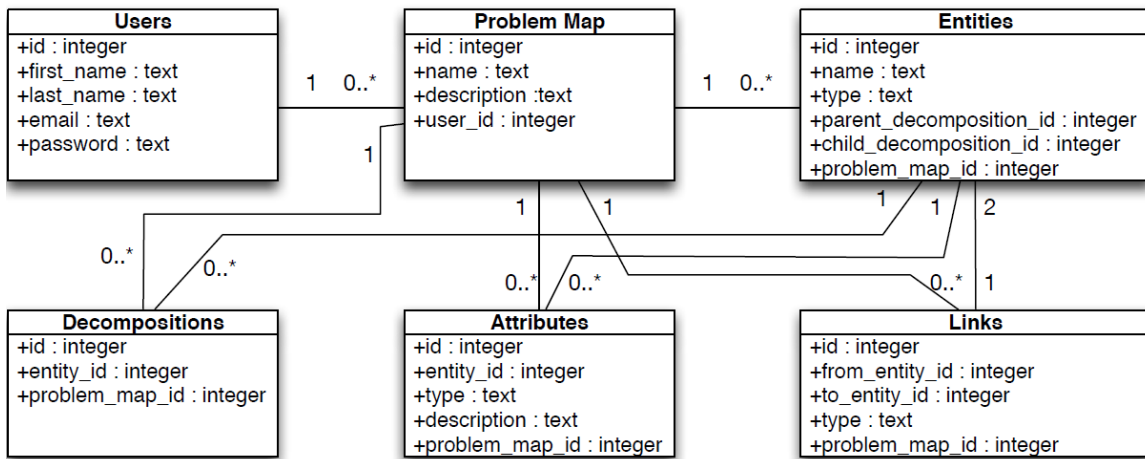


Figure 5.2 Database schema for the Problem formulator (from [96])

5.2 Graphical user interface

The tool stores problem formulation data in a relational database that reflects the problem map ontology. Once the system has stored this information, the graphical interface displays it to the user. To distinguish among the entity types, Problem Formulator presents them in separate columns. Figure 5.3 displays the main page of the graphical user interface (GUI) of the interactive testbed where formulation data is entered. At the top of the page, there are links to other views which have additional functionalities to be explained in section 5.4.

Representing individual entities is fairly trivial; the real power comes from the ability to relate these entities. Intra-group or hierarchical relations consist of alternative sets of parent-child links between entities of the same type that specify different ways to refine the parent. Inter-group relations or links consist of links between pairs of entities with different types that describe how they interact. Parent-child relations are created by drag and dropping one entity on another within the same column. Parent-child relations are shown similarly to a nested folder structure. The parent has a folder icon. The child has a file icon (if it is not a parent itself), is displayed below its parent, and is slightly indented. Alternatives (disjunctive parent-child relations) can be created by closing an existing branch and dropping the new child under the existing parent. The number of disjunctive branches is displayed above the folder.

Links are created by drag and dropping one entity on an entity in a different column. Problem Formulator's GUI displays these links by highlighting entities. When the designer mouses over an entity, the system highlights both it and all other entities that are connected to it, as Figure 5.3 illustrates.

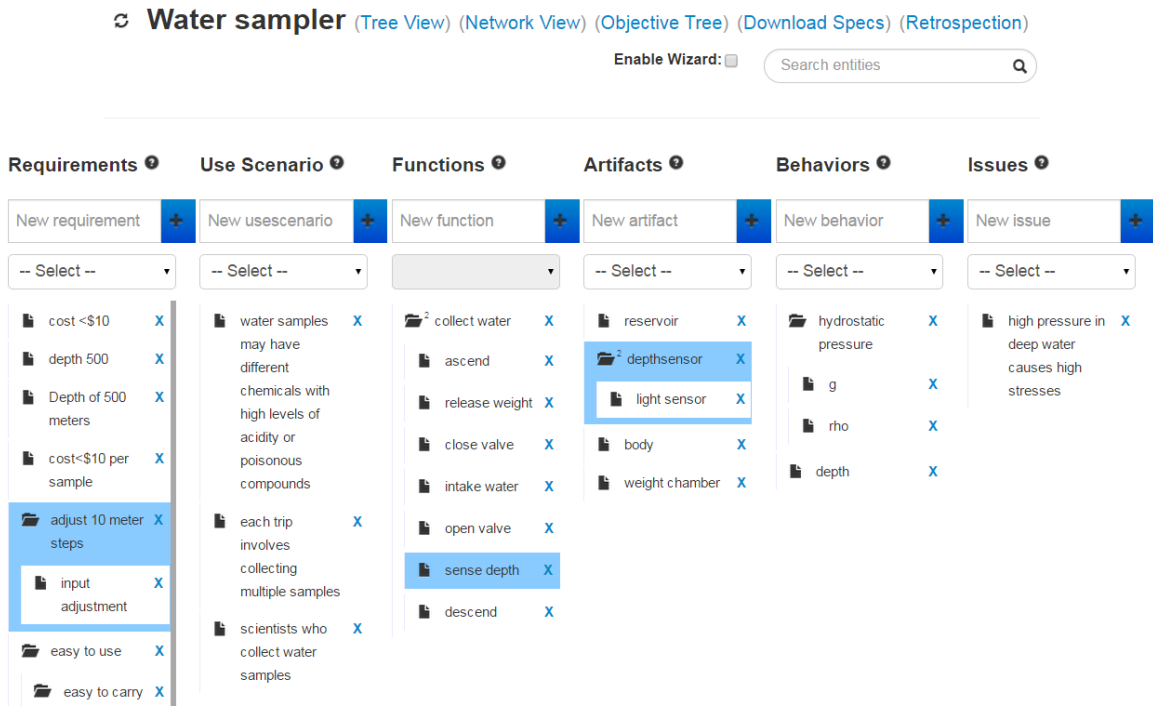


Figure 5.3 The main GUI of the Problem Formulator

5.3 Test and user studies

The design of the interface for Problem Formulator has gone through some changes throughout its development. The major change in the GUI was to move from a central node-link view (as seen in Figure 5.4) to the current multicolumn folder list view. The current version has auxiliary views as additional features which will be described in the next section. However, the main page for users to enter the data was changed to the current form.

Changes to the tool were based on a pilot user study with 11 participants. Participants were given a demo on how the tool worked with a working example. They were asked to work on a practice problem. This was followed by a survey. The survey consisted of a

few statements and the participants were asked to specify their agreement with the statement on a scale of 1 to 10 (1 strongly disagree, 10 strongly agree). When asked if the participant had experience with a similar tool, one participant mentioned FunctionCAD [97] and DANE [98] while another mentioned Concept maps [71]. The statements and responses are given in Table 5.1.

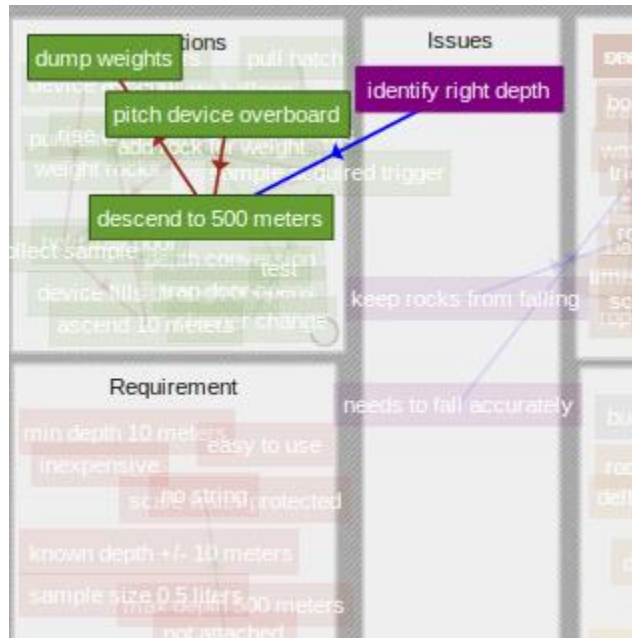
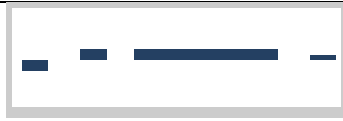







Figure 5.4 The first GUI for Problem Formulator

Though the number of participants was small, a few lessons were drawn from the responses of the survey. A demo of the tool could be helpful but the definitions for the vocabulary given in the ontology required more clarifications and examples. Most users preferred a graphical representation over a textual one for data entry. The users had a neutral opinion on whether the effort was worth the trouble which meant a more user friendly interface could win them over. They also found the tool somewhat distracting, though this was mostly because of the glitches and delays due to technical problems with

the tool. This was consistent with some of the comments which the participants additionally provided.

Table 5.1 Results of a user study on the first Problem Formulator

Survey statement	Avg.	Response range
The demo was helpful in instructing me on how to use the problem map tool.	6	
The vocabulary used in the tool (e.g. function, physical embodiment, realizes, etc.) was clear to me.	5.6	
The textual display was more helpful than the graphical display.	4	
What I was able to produce was worth the effort I had to exert to produce it.	5	
I was able to be very expressive and creative while doing the activity.	4.4	
My attention was fully turned to the activity, and I forgot about the system/tool that I was using.	4.3	

When the participants were asked ‘Please tell us what you liked about this tool.’ the majority mentioned the ability to organize their thoughts within specific categories. Other remarks included visualization, easy rewriting and editing compared to pen and paper, and complementing text and graphics. When they were asked ‘Please tell us what you did not like about this tool.’ they mentioned: having little instruction on the tool; lack of tutorials; unreadable text in the nodes when zooming; difficulty in creating links through a drop-down list. The participants were also asked ‘What functionality do you think was

missing in this tool?'. Most respondents mentioned they favor an adviser or a feedback system that helped them in exploring possible designs or telling them whether they were correct. Other suggestions included: auto arranging the nodes to avoid clutter; automatic linking of entities (if entity A is linked to entity B, linking a new entity C to B should invoke an automatic link between C and A); and printing the map. The feedback from the survey participants led to changes in the GUI. The effect of some of these changes was shown in describing the main page of the existing GUI in the previous section; the multicolumn folder list, and highlighted linked nodes is in response to user complaints about clutter and confusing display of relations. The feedback from the survey also led to other improvements which are explained in the next section.

5.4 Improvements and added features

The user study conducted on the first version of Problem Formulator laid a roadmap towards making enhancements to the GUI and including additional features. Some of the complaints which the users have made throughout the development of different versions of Problem Formulator relate to the tool being slow. This is a technical difficulty which requires improving the code; it is out of the scope of this discussion. Four features were added which are in the latest version of Problem Formulator. They are: additional views for relations, outputs for documenting and communicating one's formulation, step-by-step tutorial wizards, and a retrospective module. It should be noted that these features improved Problem Formulator as a conceptual design support tool, not a data collection testbed. No claims are made about the effect of the tool on creativity.

In order to see the relations among entities more clearly, two new views were added. One is a tree view to display hierarchies and disjunctive branches more effectively. The main GUI had a similar view to a tree with files branching out of folders but disjunctive branches could not be shown. This confused the users too. One option was to show disjunctions as layers but this was technically challenging to implement. Instead, a new auxiliary view was added. In the new tree view the user can see one or more entity type by collapsing or expanding the layout accordingly, see Figure 5.5.

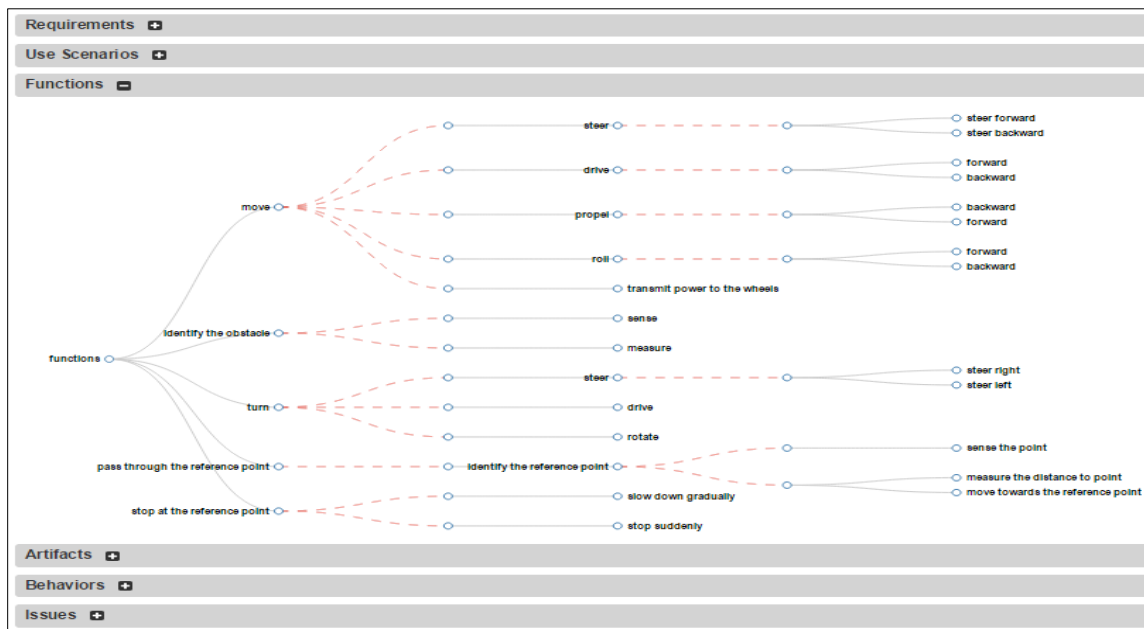


Figure 5.5 Problem Formulator enhancements - Tree view

In addition, within each group, the branches of the tree can be collapsed or expanded at every node level by clicking on the node. The conjunctions and disjunctions are distinguished by different line styles. Red dashed lines denote disjunctions (OR relation) and solid lines denote conjunctions (AND relation), see Figure 5.6. The second additional view shows the links among all entities placed around a circle, see Figure 5.7.

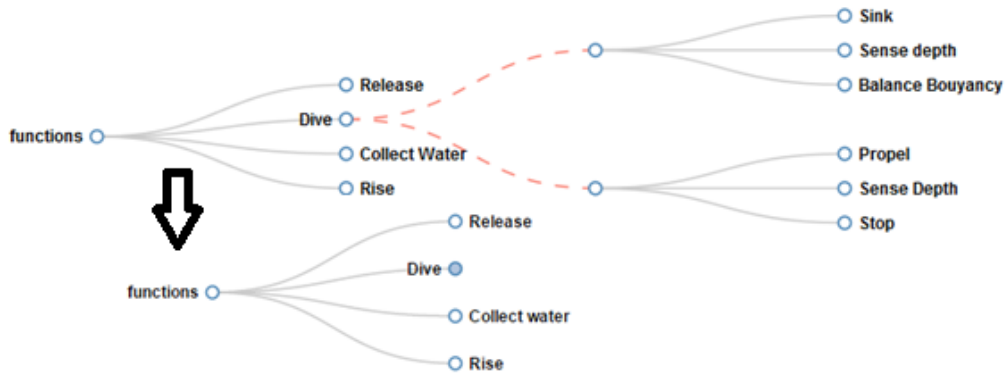


Figure 5.6 Problem Formulator enhancements – Collapsing nodes

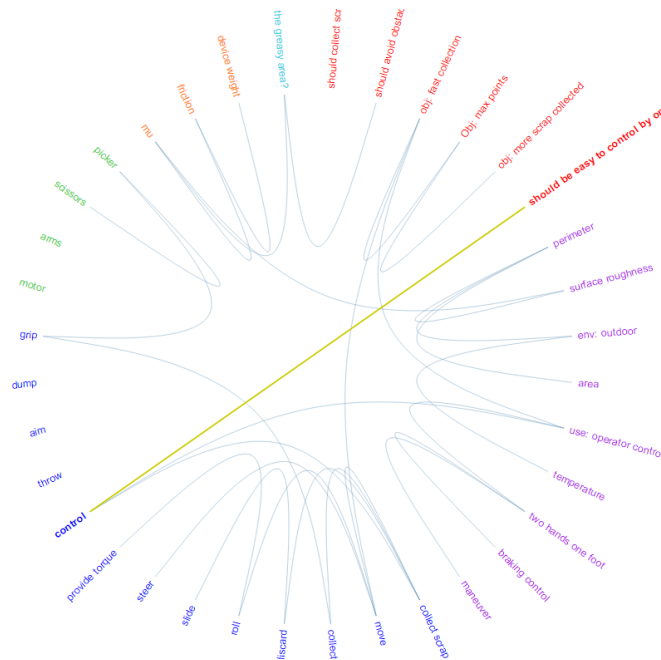


Figure 5.7 Problem Formulator enhancements – Network view

One of the drawbacks of the early versions of Problem Formulator was lack of a formal output from the tool. Users complained that the tool would be more attractive if they could save or print outputs for their own documentation or communicating their formulation with others. One common output relate to problem formulation is objective tree. To create an objective tree in Problem Formulator, first they should be specified as

subtypes within the requirements group. The objective tree structure should then be created as usual. The next step is to assign weights to the branches of each node. The weights are assigned to each node such that the sum of the weights of all children in one conjunctive branch equal to 1, see Figure 5.8. The output will be similar to Figure 5.9.

Calculate Weight
Submit
Cancel

Water sampler

Name	Weight	
cost<\$10 per sample	: 0.333	1 ▼
easy to use	: 0.667	2 ▼

easy to use

Name	Weight	
minimize setup time	: 0.200	1 ▼
store multiple samples per trip	: 0.400	2 ▼
easy to carry	: 0.400	2 ▼

easy to carry

Name	Weight	
weight less than 15 lb per bach	: 0.667	2 ▼
appropriate grip	: 0.333	1 ▼

Figure 5.8 Problem Formulator enhancements – Objective tree input

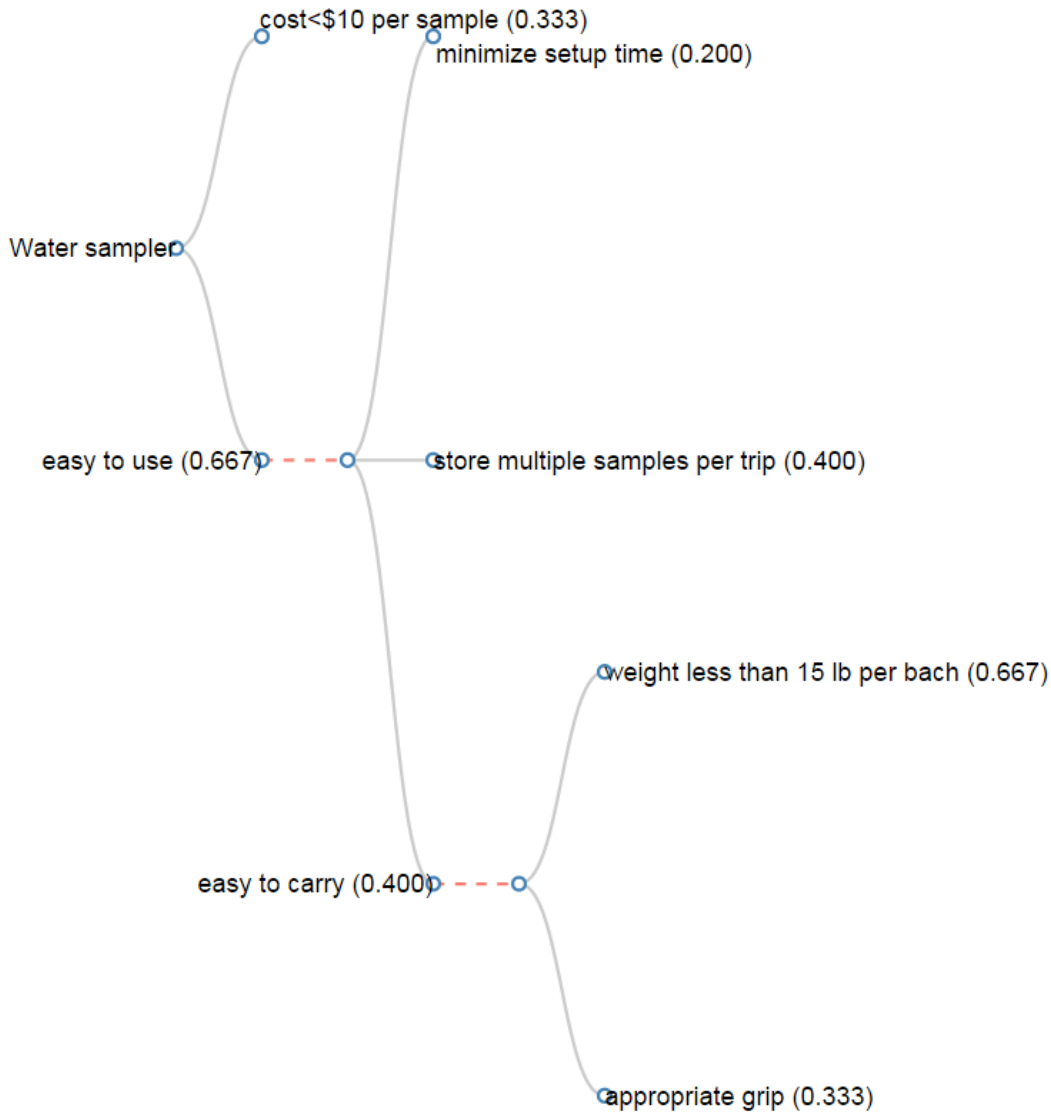


Figure 5.9 Problem Formulator enhancements – Objective tree output

The majority of the participants in the user study pointed to little instructions they received about using the tool even though they found a demo useful. They also wanted a feedback system in the tool that told them if they were correct in their formulation. In order to minimize bias towards a specific way of formulating problems, participants throughout this research have been familiarized with the definitions of the P-maps

ontology but told that they can add and edit entities and relations anyway they want. Nevertheless, users needed more instructions before they became competent in using Problem Formulator, especially help integrated within the tool. Therefore, two tutorial wizards were incorporated in Problem Formulator following two different ways of formulating problems. The two approaches have a particular process which makes them easy to follow. However, they are two out of many possibilities and it is emphasized to the participants that the wizards only show the users how to work with the tool not how to formulate problems.

The first approach is called depth expansion. The depth expansion approach tells the users to expand entities in details at lower levels as much as possible before moving to the next entity type, see Figure 5.10. The second approach is called breadth expansion. It tells the users to describe all the aspects at an abstract level in each entity before going into the details, see Figure 5.11. The wizard can be turned off once the user becomes more confident in working on his own.

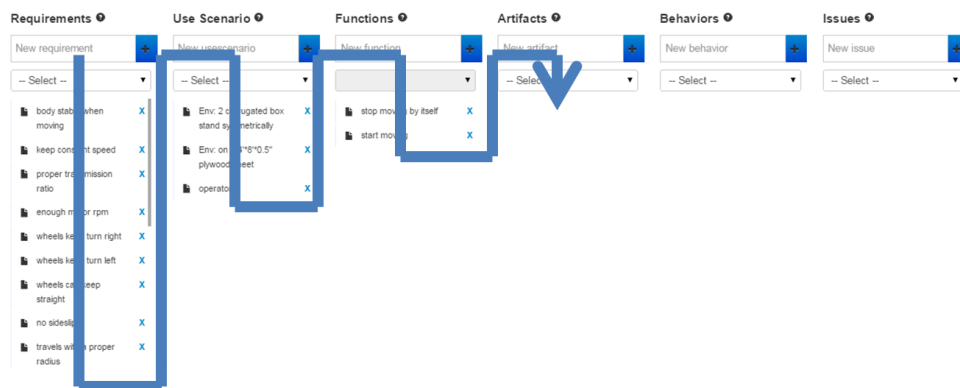


Figure 5.10 The depth exploration approach

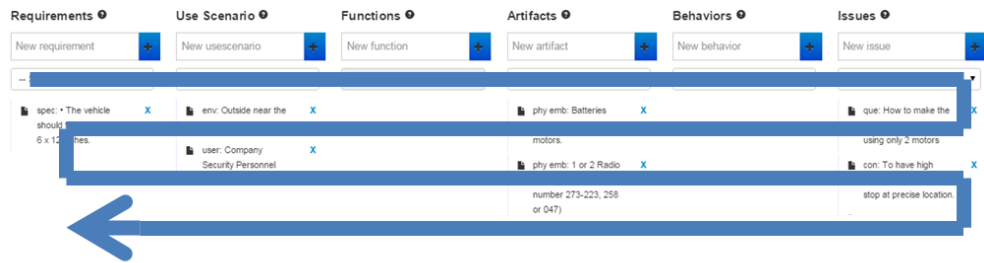


Figure 5.11 The breadth exploration approach

The last feature that has been added to Problem Formulator is a module to support retrospection. Formulating a problem in the tool is done by taking several steps such as adding an entity or linking two entities. One criticism of studying design thinking with a testbed that lacks intervention by a researcher is that the rationale of the designers' moves might not come to light. On the other hand, intervention by a researcher who asks questions about the designers' rationale might interfere with the assigned task. Even if there are no interventions by others, designers might forget to explain what they do. A solution to the dilemma caused by the tradeoff of intervention to get rationale and forgetting to express rationale is to use strength of two approaches: not to intervene while the designer is formulating the problem, and save the sequence of his moves and replay it later so he does not forget what he did. The retrospective module is shown in Figure 5.12. The designer can replay his formulation and see all the steps taken at each point. The designer can provide a response on why he took the step either from an existing list of responses or by adding a new response. This feature of Problem Formulator facilitates a new method towards research in design thinking. Since it has been added recently, it is a part of future work and therefore excluded from this dissertation.

Process Replay (List View) (Tree View) (Network View) (Objective Tree) (Download Specs)

Actions: Add | Rename | Decomposition | Link | Delete

The screenshot displays the 'Process Replay' interface. At the top, there are navigation buttons: 'Prev', 'Next', 'Play', and 'Stop'. Below these, the 'Action' is 'Requirements - Added new Entity 'obj: no leakage of batteries''. The 'Perception' is 'I had knowledge of a similar problem', and there is a 'Choose Perception:' dropdown menu. The main area is divided into six columns: 'Requirements', 'Use Scenario', 'Functions', 'Artifacts', 'Behaviors', and 'Issues'. Each column contains a list of items with icons. Below the columns is a grid of 40 small grey circles, with one circle highlighted in yellow.

Requirements	Use Scenario	Functions	Artifacts	Behaviors	Issues
obj: no leakage of batteries	user: designer of the device	pass through the reference point	phy: wires	para: size of the box/building to be circled	
obj: no harmful gases should be emitted from the device	env: All weather device	identify the reference point	phy: springs	para: force required to stop the device	
obj: components used should not chemically react with the environment	env: should be able to move on a plywood sheet of size 4'X8'X0.5'	move towards the reference point	sol: newtons laws of motion	rel: total distance covered over average speed	
obj: autonomous device		stop at the reference point	phy: sensors	para: time to comback to the reference point	
obj: device to start, stop and move automatically		stop suddenly	phy: gears	para: distance covered	
obj: the device should take less		move	sol: steering mechanism	para: average speed of the device	
		roll	phy: wood	para: weight of the device	
		identify the obstacle	phy: aluminum sheet		
		measure	phy: LEGO parts		
		time	phy: wheels		
			phy: Batteries		
			phy: 2 radio shack		

Figure 5.12 Problem Formulator enhancements – Retrospective module

Problem Formulator and its features can help designers in formulating problems. However, the main objective of creating the tool was for it to serve as a testbed; the means of easier data collection and analysis on a larger scale. Empirical studies could now be planned and executed. Preliminaries of the empirical studies are described in the next chapter.

CHAPTER 6

EMPIRICAL STUDIES-PRELIMINARIES

The previous two chapters explained the P-maps ontological framework, and the Problem Formulator testbed. Armed with the framework and the testbed, experimental studies can be conducted to answer the other research questions about differences in designers' problem formulation and its relation to creative outcome. Three experiments were carried out. The objective of the first experiment set was to show differences between experts and novices in problem formulation. The second experiment set examined the relation between problem formulation and creativity. The third experiment set tested if creativity can be predicted from problem formulation. In other words, whether the results of the second experiment could be generalized was examined. Each of the experiments is described separately in the following three chapters. A few preliminaries underlie the experiments. This chapter describes them. They are design problems or tasks, participants, and the collected data from the participants on the assigned tasks.

The collected data is on problem formulation and creativity. Problem formulation data consists of P-maps taken from coded protocols or entered in Problem Formulator. Creativity data comes from two essentially different assessment methods. One is an a priori test of creativity, i.e., it is not an assessment of one of the assigned design tasks in this study. The other is an assessment of the outcome of the design tasks at the end of the conceptual design phase.

The creativity test provides an assessment of a person's creativity within the scope of the test. Though it can show the test taker's potential creativity, it does not necessarily

reflect on the test taker's creative outcome on a design task. It is possible to score high on the test but have a poor outcome on the design task. Therefore, there is a need to separate the two assessment methods. The creativity test measures can be used as an initial evaluation of the distribution of participants within study samples with respect to conceptual design skills. They support the argument whether the participants in the sample represent a larger population of designers. Shah *et al.*'s [99] Divergent Thinking test has been used for this purpose. Shah *et al.*'s [24] ideation metrics, on the other hand, have been used to evaluate the creativity in conceptual design outcome. The experiments are summarized in Table 6.1.

Table 6.1 Summary of the design of experiments

	Experiment I	Experiment II	Experiment III
Objective	Showing differences within and between experts and novices	Understanding the relation between problem formulation and creativity	Predicting creativity from problem formulation
Input	Problem formulation characteristics	Problem formulation characteristics Ideation metrics	Formulation-ideation models Ideation metrics
Output	Differences in formulation characteristics	Models of ideation vs. formulation Differences in ideation metrics	Differences between predicted and actual ideation

6.1 Characteristics of design problems

There are a few criteria for choosing an appropriate design problem for this study. One is that the problem should not be too technical for the subject designers, i.e., it should not require extensive domain knowledge to understand the problem and come up with a design solution. Second, the problem should lead to diverse solutions. The level of

difficulty should be in such a way that different designers propose a variety of solutions. This affects the range of outcome ideation metrics as the dependent variable in the second and third experiments. Third, the problem should have some conflicting requirements and key challenges similar to many real-life design problems. These criteria are similar to what Dixon *et al.* [100] refer to as a novel problem. Most of the selected problems come from engineering design course books.

Five design problems were used in this research. To avoid repeating their names or descriptions in each experiment, they are described here but will be referred to with a code from DP_1 to DP_5. The first design problem (DP_1), the water sampler, is about designing a water collection device for taking fresh water samples from lakes. The problem is taken from Pahl and Beitz [31]. The given problem statement is in Figure 6.1. The second design problem (DP_2), the can crusher, is about designing a device that discards aluminum cans. The problem statement is given in Figure 6.2.

Design a mechanical device to be used from a rowboat by a researcher who wishes to collect samples of water from fresh- water lakes (e.g., Lake Tahoe) at known depths down to a maximum of 500 m. After release, the device must not be attached to the boat and must descend to within 10 m of an easily adjustable pre-determined depth. It must return to the surface with a 0.5-liter sample of water from that depth and then float on the surface until picked up. The device should be reliable, easy to use, reusable, and inexpensive.

Figure 6.1 Problem statement for ‘water sampler’ (DP_1)

Design a machine to accept and store used aluminum drink cans for subsequent recycling. The device is to be located in busy public areas and is to accept cans one at a time from an individual and pay out a coin as a reward. To reduce storage space, the can is to be crushed to a height of approximately 15 mm. The maximum crushing force required is 2 kN. The original height of a can lies between 115 and 155 mm, a typical diameter is 65 mm and the average can mass is 0.02 kg.

Figure 6.2 Problem statement for ‘can crusher’ (DP_2)

The third problem (DP_3), the goofy gopher, is to design a device that collects more golf balls than an opponent's device and stores them in the respective silos. Balls of different color have different points, see Figure 6.3. Stealing balls from the opponents and interfering with the operation of their devices is allowed.

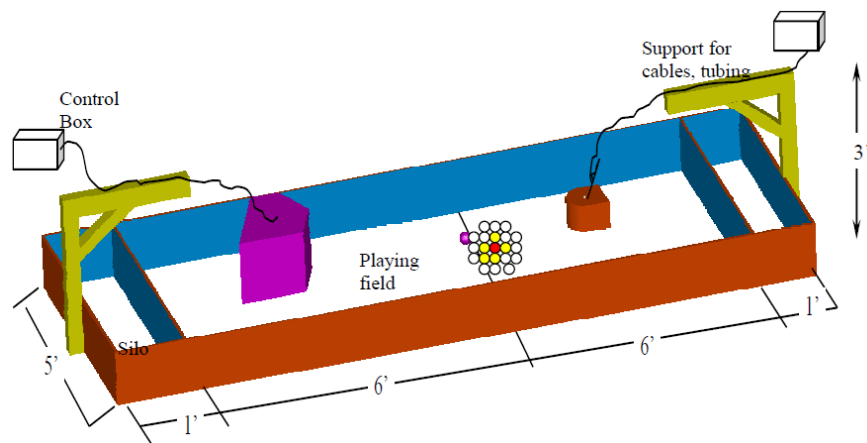


Figure 6.3. The settings for the ‘goofy gopher’ problem (DP_3)

The fourth problem used in this research (DP_4) is to design the shot buddy (taken from [76]); a device which returns shot basketballs to the shooter, whether the basket was made or missed, see Figure 6.4. The problem statement also says that the device must be able to automatically adjust the return angle based on the position of the shooter when the ball is shot. It must also accurately and quickly return balls to the shooter and not block the shooters access to the basket. Ideally, the return speed would be adjustable to accommodate different skill levels. The device should be user friendly for kids ages 10-18, easy to setup and applicable to a wide variety of basket types. The device should be affordable for the average family.

The last design problem (DP_5) is to design an autonomous surveillance vehicle to automatically and periodically tour the perimeter of two structures, stopping as close as possible to the start point, see Figure 6.5.

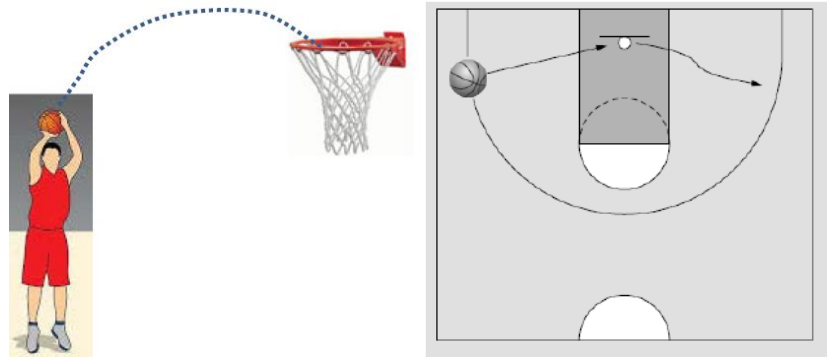


Figure 6.4. The settings for the ‘shot buddy’ problem (DP_4)

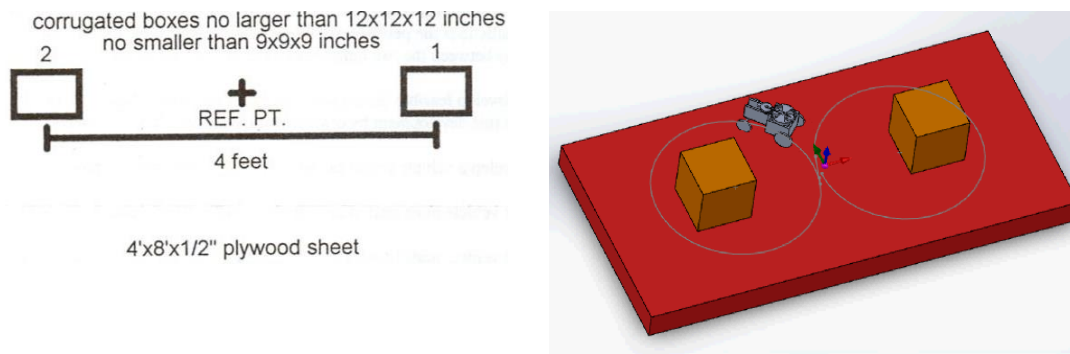


Figure 6.5. The settings for the ‘autonomous surveillance’ problem (DP_5)

6.2 Characteristics of participating designers

To meet the objectives of the three designed experiments, two characteristics of the participating designers in this research should be taken into account. One is expertise and the other is creativity in conceptual design. There are two levels of expertise considered in this research: expert or novice. This is a common consideration in research in designer thinking [2]. Many studies of designer thinking compare experts to novices; they do not define expertise with years of experience as a numerical variable (neither does this study). Unlike expertise, the second characteristic (creativity) is defined with a set of numerical metrics. As explained in the introduction of this chapter, there are two measures of creativity: apriori Divergent Thinking test scores, and aposteriori ideation metrics. The following discussion about the type of variable holds true for both characterizations of creativity.

Both measures characterize a participating designer’s creativity. The main intention behind separating the two is that the former is a potential measure which can be used for selecting the appropriate participants for the designed experiment. On the other hand, ideation metrics are characterizations of the designers’ actual outcome. It is possible to

treat creativity as a class variable by setting a threshold value above which the designer is considered creative. Except for a few classifier models (for a posteriori ideation metrics) discussed for experiment II, creativity is characterized as a numerical variable. There are two reasons. One is that both the test and the outcome consist of several independent measures. It is less likely to find one designer who excels on all measures to label him or her creative, much less many designers. The other reason is the limited resources for the experiments with regard to recruiting participants. The creativity test scores could not be used to screen creative and non-creative participants from a larger pool. There were not many participants to recruit, and conducting the a priori test for a large pool would be resource consuming. Instead, the creativity test scores serve two purposes. First, they provide a basis for determining how well the recruited participants represent the population of designers. This can be done by comparing participants' test scores to a large historic sample. Second, they can be used for tracking participants' evolution in becoming more creative. This can be achieved by following the change from participants' potential creativity determined by the test to the actual creativity determined by the ideation metrics on the outcome of assigned design tasks.

The Divergent Thinking test [99] has four direct and four indirect measures. The four direct measures are fluency, flexibility, originality, and quality. The indirect measures relate more to cognitive processes. Therefore they are not considered here. The direct measures on the other hand relate more to outcome. They correspond to the ideation metrics defined in [24]. Fluency, flexibility, originality, and quality correspond to quantity, variety, novelty, and quality of ideas respectively.

For the three experiments in this research, four groups of designers participated in the study from the fall of 2011 through fall of 2014. There was one group of experts and 3 groups of novice students. The first group from fall of 2011 (labeled F11E) consisted of eight expert designers from the industry (a consumer electronics company). The second group of participants was about sixty students of an undergrad mechanical design course. The third and the fourth group of participants were mechanical engineering graduate students of an advanced product design course during the fall of 2013 (F13G) and 2014 (F14G). The apriori assessment of the participants' creativity (with the Divergent Thinking test [99]) was done for all groups except the undergrad students (F12U); conducting and scoring the test was unfeasible for the sixty undergrad participants. In addition, the sample was large enough to ensure having a normal distribution in the sample (more than 30 participants). The results of the Divergent Thinking test scores are shown in Table 6.2. Except for max originality, the experts had a slightly higher score compared to the students (though with a narrower distribution with a 0.59 STD). Overall, the scores of originality and quality are closer to a historic sample compared to fluency and flexibility; see Figure 6.6.

Table 6.2. Distribution of Participants' Divergent Thinking test [6] scores

Test component	Mean			STD			Min-Max		
	F11E	F13G	F14G	F11E	F13G	F14G	F11E	F13G	F14G
Fluency	5.9	3.2	3.3	1.13	0.78	1.07	4.5-7.5	2-5	1.5-5.5
Flexibility	5.5	3.7	3.6	0.7	0.92	1.01	4.4-6.8	2.5-6	2-6.3
Originality	5.2	4.3	4.2	0.8	0.77	0.74	4-6.3	2.4-5.5	3-5.6
Max orig.	7.5	7.1	7.4	0.59	1.4	1.45	6.4-8.1	4.1-9.8	4.2-9.9
Quality	6.6	5.2	5.2	1.66	1.46	1.58	4.3-9.3	2.4-8	2-8

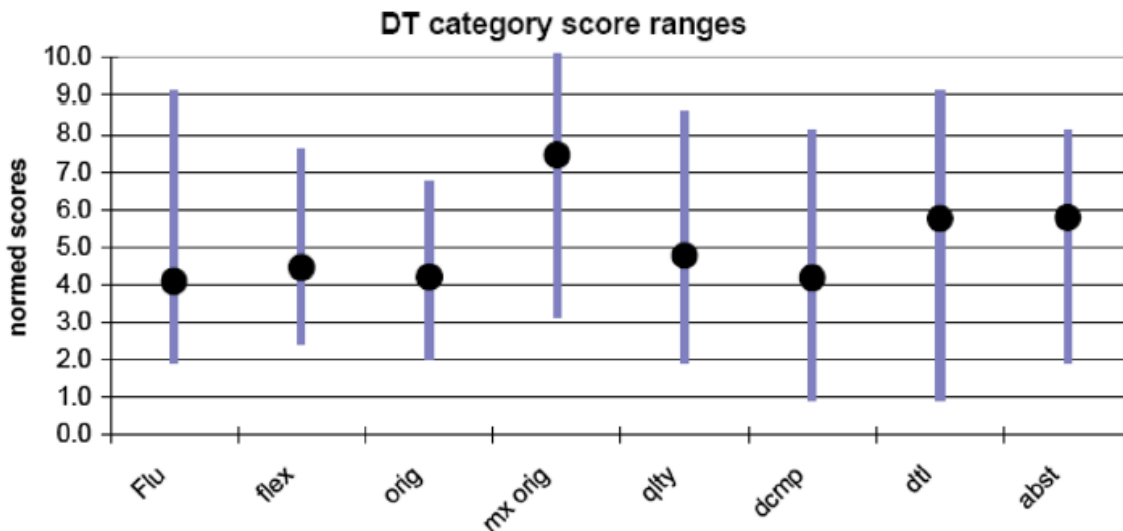


Figure 6.6 A historic sample of the Divergent Thinking test scores (from [99])

6.3 Characteristics of problem formulation

The characteristics of problem formulation are defined based on the P-maps ontology. Different variables can be extracted from P-maps. There are two different ways to define characteristics of problem definition expressed in P-maps. One is to define characteristics of a state, and one is to define that of changes across states obeying certain conditions.

6.3.1 State characteristics

State characteristics can be defined as characteristics of accumulated data fragments over a time period up to a point, the state. Therefore, the characteristic can be computed by looking at that state alone, regardless of previous states. A simple example is the number of instances of an entity such as requirements. To find this characteristic, one does not need to know how the process was, e.g., whether most of the requirements were

added earlier in the process or if they were connected to other entities. Examples of different state characteristics are given in Table 6.3.

Simple counts determine the number of instances of one type of entity such as functions. The proportion characteristic is a normalized version of the simple counts and can be useful in removing the effect of the design problem which often leads to different numbers of expressed entities when the same designers work on different problems. Another characteristic, isolated entities such as isolated artifacts, is the count of entities which are not a part of any hierarchy within each group. This characteristic may refer to unrelated fragments or ones which are independent of each other at a high level of abstraction within an aspect of the problem since they are not further decomposed.

Table 6.3 Examples of P-maps state characteristics

Characteristic	Description
Simple count	Total instances of one entity
Proportion	Proportion of instances of one entity to total instances
Isolated entities	The number of entities in each category that are orphan, i.e., entities with no parents and children within a group
Disconnected entities	The number of entities without any relation to entities in other categories
Inter-group links	The number of links between any two types of entities
Intergroup alternatives	The sum of all alternatives for all the nodes in an entity
Hierarchy depth	Maximum number of levels of hierarchy for each entity
Deepest entity	The entity with the maximum hierarchy depth
Average alternatives	The average number of alternatives (disjunctive branches) under a node

Number of disconnected entities, e.g., disconnected functions, is the number of entities within each category which are not related to other categories. Such a characteristic can relate to the inability in understanding the relationships among different aspect of the problem formulation. A designer may consider different environmental or usability factors that affect a given design problem (high number of use scenarios), but fail to identify how these factors situate the requirements or what interactions (affordances) are at play in the proposed artifacts. On the other hand, the number of inter-group links and intergroup alternatives highlight the relationships that the designer finds among different categories, and the different number of ways problem formulation fragments relate respectively. Hierarchy depth and the deepest entity characteristics give an idea about what aspects of the problem formulation the designer focuses on at a state in time.

The proposed characteristics may seem to correlate or co-vary which one should take into account when studying them in relation to other dependent variables. However, the characteristics by definition are not correlated. For example, one might suggest that the number of disconnected entities co-vary with the number of inter-group links. This is not always true. For example, there can be a few requirements that are disconnected but the number of links is high between requirements and functions and few or non-existent between requirements and the other five categories. Another example is hierarchy depth and the deepest entity characteristics. In comparing two different designers at the same state (after the same amount of time spent or after the same number of actions taken), one might find that both have decomposed the functions to four levels but the deepest entity for one is functions while the other has expanded the requirements entity the furthest.

Finally, the average alternative characteristic can show the number of proposed alternatives per node in any category.

6.3.2 Examples of state characteristics

To show how these characteristics can be computed let us consider an example of a P-map of a problem where one should design a human-operated device which collects scrap from a field. Since the Problem Formulator testbed provides multiple views of the data, it is convenient to describe the example through those views. Figure 6.7 shows a snapshot of a P-map state taken from the main GUI of the testbed. The tree view in Figure 6.8 provides a clearer way to show the disjunctions. Inter-group links are easier to count in the network view of Figure 6.9. With these views, some state characteristics for this example can be counted. The main GUI snapshot shows the total number of requirements to be 6. There are 3 requirements which are not in relation to other requirements, hence isolated. The number of disconnected requirements (with no links to other entity types) can be found from the network view; it is 3.

It can be seen that the number of isolated and disconnected entities are not related to each other. The requirement “should collect scrap” is both isolated and disconnected; “should avoid obstacles” is isolated but not disconnected from other entities (it is related to an issue); “obj: max points” is not isolated but disconnected. Figure 6.9 also shows that there are two links between the requirement and the function categories, one of which is highlighted in the figure. From the first snapshot it is easy to find that use scenarios have a hierarchy depth of 3 while functions are the deepest entity. To compute the average number of proposed alternative function decompositions at the second level of the

function hierarchy one can see Figure 6.8 where there are 6 disjunctive branches under the 4 nodes. A summary of some characteristics is in Table 6.4.

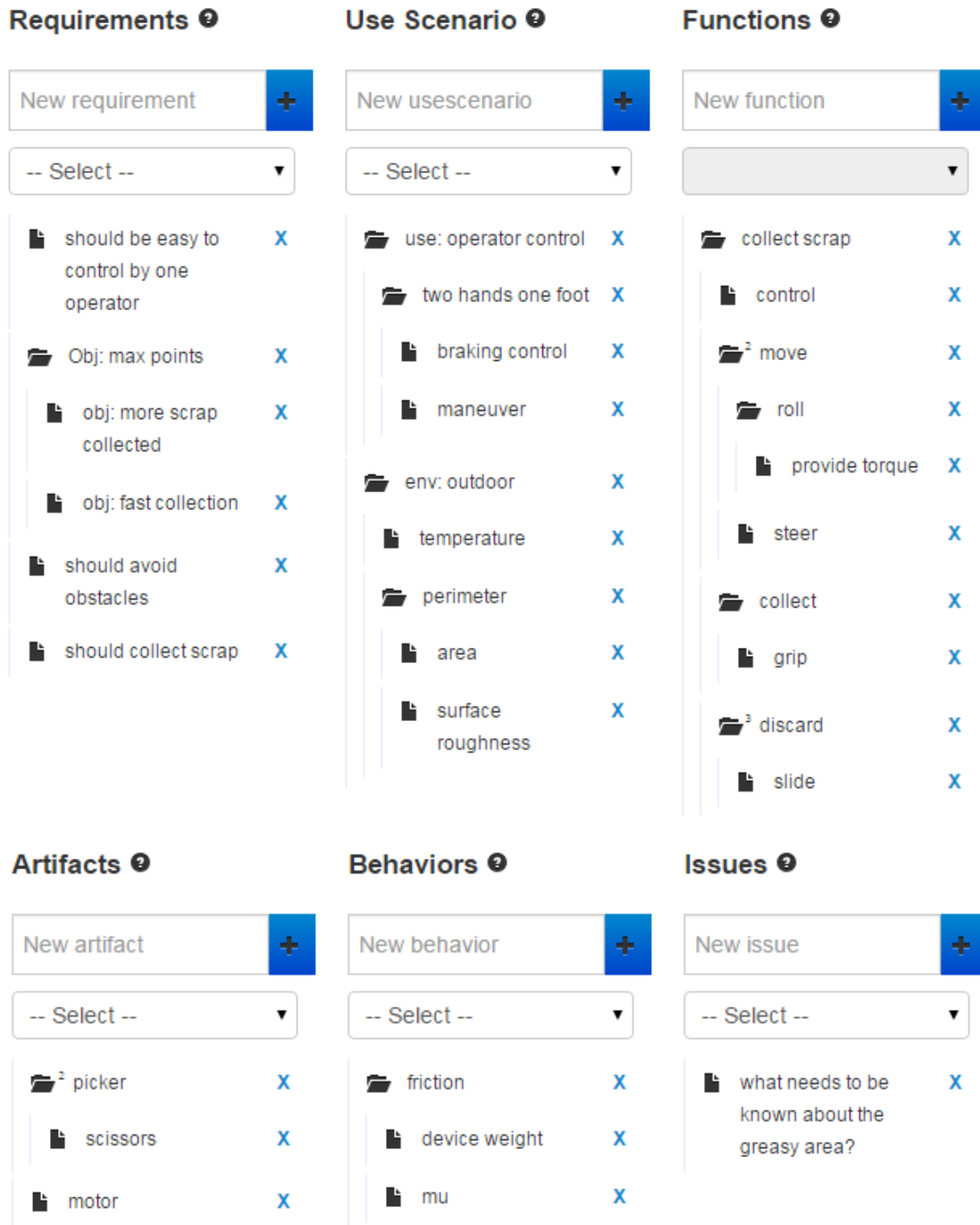


Figure 6.7 A snapshot of a P-map for the state counts example

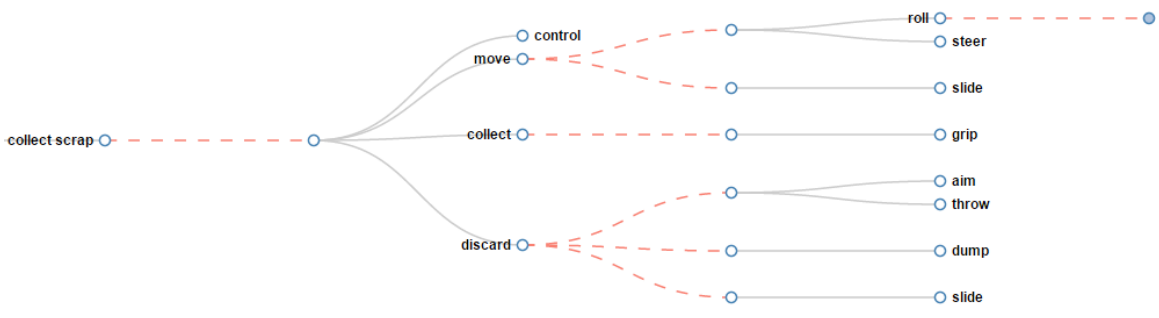


Figure 6.8 Tree view for the state counts example

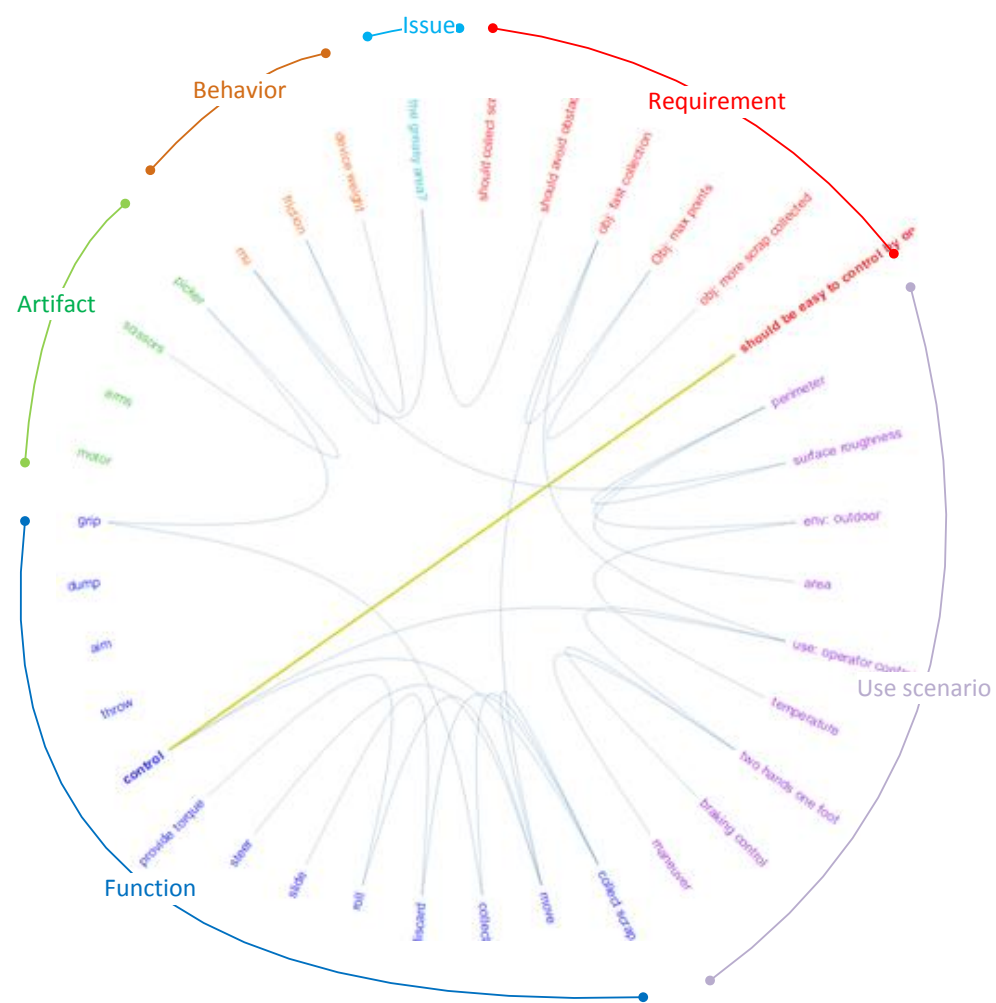


Figure 6.9 Network view of the state counts example

Table 6.4 An example of state counts for a P-map

Characteristic	Example	Value
Simple count	Requirements	6
Isolated entities	Isolated requirements	3
Isolated entities	Isolated functions	0
Disconnected entities	Disconnected requirements	3
Disconnected entities	Disconnected functions	3
Hierarchy depth	Use scenario depth	3
Deepest entity	Deepest entity	<i>function</i>
Inter-group links	Requirement-Function links	2
Average alternatives	Average alternative functions (at the second level)	6/4

6.3.3 Process characteristics (strategies)

The second type of characteristic deals with specific changes across states, representing a pattern often corresponding to a strategic move. The emphasis on the specificity of the changes is because one can in a way define the state characteristics which were presented in the previous chapter as process characteristics too, though between a blank start state and the state which is being measured. The number of functions as a state characteristic can be assumed as the changes in the number of functions from the start state. There are also characteristics that relate to changes in time but are not necessarily representing a strategy. Consider a sequence of different entity types added in a P-map, e.g., ‘requirement, function, requirement, artifact, function, function’ and a timestamp assigned to them based on the order of addition, i.e., 1 through 6. A variable can be defined as the median of occurrences of an entity. In the given

sequence, requirements are added at times 1 and 3, and functions are added at times 2, 5, and 6, thus the median of occurrences of requirements and functions are 2 and 5 respectively; there are even number of requirements and odd number of functions, therefore, median of 1 and 3 is 2, and median of 2, 5, and 6 is 5. This is a process characteristic which does not represent a specific strategy.

Number of occurrences of a strategy is a characteristic of problem formulation. P-maps can be used to represent and formalize strategies that designers adopt. A general description of a formalized strategy is defined by a set of conditions that occur across states during the development of P-maps. The strategies which are looked for are chosen based on the hypotheses that will be examined in the experiments. They are identified in the literature relating to creativity or expertise. One strategy is abstraction in problem definition. When defining a problem, a designer can add more detail to a fragment or entity, or generalize it. The ability to abstract concepts is considered a key in creative design [21]. To see whether a designer has employed an abstraction strategy during an interval, one can state the conditions as the changes within the interval. For the abstraction strategy, the conditions can be stated as if: a) entity E1 added at time T1, b) entity E2 added at time T2, c) E2 is a child of E1, and d) E1 is added after E2 or $T1 > T2$. Other strategies relate to: exploring entities in depth rather than breadth [39]; following a systematic order in decomposing different aspects of the problem either as it is observed in practice [38] or as prescribed in design textbooks [31]; identifying conflicts and tradeoffs [77]. These strategies are defined in Table 6.5. The two strategies *order req_use* and *order req_fun* are similar to the *forward order* but at a micro-level. They are considered specifically because they focus on problem-oriented aspects of P-maps

entities. Strategies *order req_use*, *order req_fun*, and *forward order* relate to testing hypothesis H1_a which compares experts to novices in exploring problem definition; novices are expected to be systematic while experts are expected to be opportunistic. The entity depth prevalence relates to testing hypothesis H2_a which states that depth-first exploration leads to more creativity. It should also be noted that problem formulation strategies are not limited to the ones identified so far. There can be many other strategies. Search, proposition, and formalization of new strategies is a part of future work which will be discussed in section 11.3.

Table 6.5 List of formalized problem formulation strategies

Strategy	Definition	Conditions
Abstraction	The designer refers to a more general aspect at a higher level	Entity parent added at time t1 Entity child added at time t2 $t1 > t2$
Entity depth prevalence	The designer develops details of an aspect in depth before relating it to other categories	Entity parent of type A added at time t1 Entity child of type A added at time t2 Entity of type B added at time t3 Entity of type B related to parent entity of type A at time t4 $t4 > t2$
Forward order	The designer follows a specific order from requirements to issues	Any subset of entities requirement, use scenario, function, artifact, behavior, issue are added at time t1 through t6 Any pair of added entities is linked at time t7, t8, etc. $t1 < t2 \dots < t6$ $t6 < t7 < t8 \dots$
Order req_use	The designer follows a specific order adding use scenarios after all related requirements	A requirement is added at time t1 A use scenario is related to the requirement at time t2 Entity of other type added at time t3 and related to the requirement at time t4 $t2 < t3 < t4$
Order req_fun	The designer follows a specific order adding functions after all related requirements	A requirement is added at time t1 A function is related to the requirement at time t2 Entity of other type added at time t3 and related to the requirement at time t4 $t2 < t3 < t4$
Conflict identification	The designer identifies an issue about conflicting requirements	Requirement R1 is added at time t1 Requirement R2 is added at time t2 Issue I1 is added at time t3 I1 is related to R1 and R2

6.3.4 Examples of strategies

To clarify how these strategies are found in P-maps the mechanism of tracing their occurrences should be explained. To trace occurrences of strategies, first P-maps fragments are written as predicates (logic statements). Second, the strategy is formally declared as a set of logical statements that has certain conditions. Third, an Answer Set Programming (ASP) [101, 102] grounder/solver is used to trace occurrences of strategies by finding matches for the conditions among the P-maps predicates. The reasons for choosing ASP are:

- Ease of analysis in a declarative syntax compared to procedural programming.
- Simplicity of the logical formalism that makes encoded fragments easily readable and close to natural language.
- Ease of performing automated reasoning over the P-maps predicates.
- Easy conversion of P-maps data from a conventional database to an ASP representation.

Answer set programs consist of two main components: facts, which are the ground literals over which the system reasons, and rules, which are used to perform logical reasoning over the facts. Predicates are represented with a name followed by braces which contain the values of the attributes that define the predicate. P-maps data fragments can be easily represented as predicates. The requirement ‘should collect scrap’ in the P-maps shown in Figure 6.7 can be represented as the predicate `requirement(should_collect_scrap,1)` where 1 shows the time when the requirement was added.

To explain the tracing mechanism let us introduce a definition of states in the P-maps framework. The definition may seem to be arbitrary considering the fact that it is difficult to clearly define boundaries of mental states for human subjects. Consider the simple case where any change such as the addition of a new instance of an entity, specifying an attribute of an existing entity, or relating two instances is an operator that alters the current state into a new state. Strategies can be traced by comparing two states in an interval during which one expects the strategy to be employed. Going back to the example of the abstraction strategy, one can look for the states that include parent-child relations. The states that contain the parent, the child, and the parent-child relation are located. If the state that has the parent occurs after the state that has the child, it indicates that the designer followed an abstraction strategy. The representation of each state as a predicate will be:

State at T1: requirement(rq1,t1).

State at T2: requirement(rq2,t2).

State at T3: parent_child(rq2,rq1,hy1).

where $T1 < T2$ or $t1 < t2$. Instances of strategies are traced using an ASP solver program. The Potassco ASP solver [103] is used in this work. In most ASP solvers, a predicate that ends with a dot represents a fact, the head of a rule is separated from the body by colon and dash, and variables are capitalized while instances are in lower case. The abstraction strategy that was previously illustrated can be traced by using an ASP solver and applying the following rule to all the predicates (facts) that are derived from a P-map:

```
strategy(abstraction,Entity_parent):- entity(Entity_parent,T_parent),
entity(Entity_child,T_child),
parent_of(Entity_parent,Entity_child,T_parent_of), T_parent>T_child.
```

The rule matches against a parent entity whose creation is later than that of its child. For any entity that matches against the rule, an answer is generated with a predicate “strategy(abstraction,Entity_parent)”. Tracing the *forward order* strategy requires a more complex set of rules. To formally state the strategy with respect to P-maps one should look at each requirement to see if it is situated by a use scenario before being related to other entities. One should include all possible combinations of relations for this strategy (depending on what relations exist between a requirement and other entities). Two possible combinations are shown in Figure 6.10.

The ASP rules for all the strategies of Table 6.5 can be found in the appendices (Appendix B). The number of occurrences of each of the defined strategies provides a set of P-maps variables. Earlier in this section, state counts were defined. The two types of characteristics identified in this section, i.e., state counts and counts of occurrences of strategies are the problem formulation characteristics which will be used as the input variables to the experiments. The next section describes the characteristics of creative.


```

strategy(forward_order,Requirement):-
situates(Usescenario,Requirement,T_situates),
satisfies(Function,Requirement,T_satisfies),
fulfills(Artifact,Requirement,T_fulfills),
manages(Behavior,Requirement,T_manages),
relates(Issue,Requirement,T_related),
T_situates <T_satisfies , T_situates<T_fulfills,
T_situates<T_manages, T_situates<T_related.

```

```

strategy(forward_order,Requirement):-
situates(Usescenario,Requirement,T_situates),
fulfills(Artifact,Requirement,T_fulfills),
manages(Behavior,Requirement,T_manages),
T_situates<T_fulfills, T_situates<T_manages.

```

Figure 6.10 ASP encoding of the forward order strategy

6.4 Characteristics of creative outcome

The last part to define before describing the design of experiments is the characteristics of creative outcome. Earlier in this chapter a distinction was made between two sources of creativity data: the apriori Divergent Thinking test scores, the aposteriori ideation metrics. This section describes how the ideation metrics are found in the data as a characteristic of creative outcome. Before explaining this process it is necessary to provide an operational definition of creativity to justify the methods of creativity assessment used in this research. First, most definitions of creativity are related to creative outcome. As it was explained in chapter 1, it has been an accepted notion to evaluate a person's creativity by evaluating a measure of outcome. Amabile [5] introduced consensual assessment as an appropriate way of measuring creativity. She

argued that an aggregate of several judges' subjective assessment can be used to measure creativity. Second, the majority of definitions of creativity deem an idea creative if it is both novel and feasible; if an idea is novel but impractical it cannot be considered creative, nor is it creative if it can be carried out but lacks originality. However, originality can be framed in reference to the person or to history. Boden [6] called it Psychological creativity if a person comes up with an idea that is new to the person regardless of how many people have had that idea before. Historical creativity on the other hand happens when a person comes up with an idea that is globally unprecedented. Third, according to Csikszentmihalyi [104], creativity should be recognized and validated by different experts within a domain as a culture with symbolic rules. Finally, Ward *et al.* [105] describe creativity as a continuum not a discrete event. Creativity does not stop with one idea, and thus cannot be measured without considering the different ideas that a person thinks about and expresses in solving a problem. Considering these points, the following is my operational definition of creativity:

“Creativity in design relates to the ability of the designer to come up with as many ideas as possible that are not similar to each other, are new to the person and the surrounding community, are feasible, and are recognized as such by more than one expert.”

The ideation metrics of Shah *et al.* [10] are well-established in design research which underlie the definition of creativity provided above. They consist of quantity, variety, novelty, and quality. Quantity measures the total number of generated ideas. Generated ideas, especially when the problem is decomposed into multiple sub-problems might have overlaps and duplicates for some sub-problems. Variety takes into account similarity of

generated ideas. Novelty is a measure of how rare generated ideas are. It is measured in comparison to ideas generated by others in the same study sample or in a historic pool. Quality measures whether an idea is feasible or if it meets the design requirements.

To calculate these measures, the design is decomposed into its desired key functions. Weights can be assigned to each function. Every generated idea is evaluated with respect to the key functions and the solution for each function is described. If the solution for a function is similar to a previously identified idea, the same description or name is used. The collection of all the ideas gathered in this manner from all participants creates an inventory of solutions for key functions. Quantity will be the total number of ideas for all functions found by a participant. Variety will be the total number of unique ideas for all functions found by a participant. To find novelty, first all unique ideas found by each participant for each function are counted. Then the number of participants who specified a solution for a function is counted. A novelty score for each function is found by determining how rare the idea is, i.e., if all participants have the idea, the novelty score for that idea is the lowest; if only one participants has the idea, the novelty score for that idea is the highest. The novelty score for a design is the sum or weighted sum of the novelty scores of all functions. The novelty score of the participant is the average of novelty scores of all generated ideas by the participant. Quality can be assessed by a panel of expert judges who assign a score to each idea generated for each function. The quality score for a design is the sum or weighted sum of the quality scores of all functions. The final quality score of a participant is the average of quality scores of all ideas.

The described procedure was done for the experiments II and III involving finding creative outcome characteristics. Since the same procedure was used in the related design problems an example is provided in this section to avoid repeating the same procedure in both experiments (each of which involving several design problems). The example is for the goofy gopher problem (DP_3) introduced in section 6.1. In this study, the data for calculating ideation metrics came from sketches collected for each problem. A sample sketch for DP_3 can be seen in Figure 6.11. A panel of three judges chose the desired key functions as follows with the corresponding weights: move 0.2, aim 0.2, collect 0.4, store 0.05, score 0.1, and interfere or block 0.05. Four wheels, single collection and continuous dumping, scoop, platform, high ramp, and using suction to hold the opponent are the descriptions given for each of the aforementioned functions found in the sketch in Figure 6.11.

Inventories of concepts were created for each problem from a union of the solutions found from participant's sketches as explained above. Table 6.6 shows a sample from the inventory for the DP_3. In this sample inventory for two participants, it can be seen that both have 'scoop' as a solution for collect. This means that this idea has the lowest novelty score. In contrast, participant B has two unique ideas (vacuum and gripper). Since collect has the highest weight, the novelty score for each of the two ideas and in turn the final novelty score for participant B will be higher. Quantity and variety can also be found for each participant. Participant A has a total of 11 ideas for the 6 functions, 10 of which are unique. Participant B has a total of 25 ideas, 15 of which are unique. All scores are normalized on a scale of 1-10. Therefore, for the given inventory, quantity and variety scores are 4.4 and 10 for participant A, and are 10 and 6.6 for participant B.

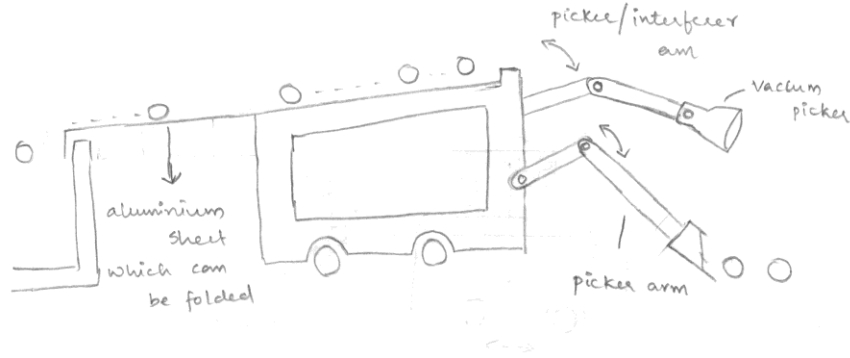


Figure 6.11 A sketch of a concept solution for the goofy gopher problem (DP_3)

Table 6.6. A sample concept inventory for the Gopher problem (DP_3)

Partic.	Move	Aim	Collect	Store	Score	Interfere/block
A	4 wheels	Single collection continuous score	Scoop	Platform	High ramp	Suction holds opponent
A	Tracks	Single collect and score	Vacuum	Platform	Conveyor	-
B	4 wheels	Single collect and score	Gripper	Platform	Elevator	-
B	4 wheels	Single collect and score	Scoop	-	Catapult	-
B	4 wheels	Single collection continuous score	Scoop	Platform	High ramp	Suction holds opponent
B	-	Multi collect and score	Sweeper	Box	Blow	Blow balls away from opponent

6.5 Design of experiments

In search for answers to the research questions a few hypotheses were stated. To test the stated hypotheses three sets of experiments were designed. Before explaining the design of experiments let us review the stated hypotheses:

H1_a) Novice designers follow a systematic order in expressing problem formulation while experts have a more opportunistic behavior.

H1_b) Experts find key issues early on during problem formulation while novices find more issues and later in the formulation process.

H2_a) Depth-first exploration of problem formulation entities leads to more creativity.

H2_b) Creativity can be improved in novice designers by teaching them characteristics of good problem formulation.

H3) Creativity in design outcome can be predicted with an acceptable degree of confidence from problem formulation behavior.

Each of the experiments examines one or two of the hypotheses. The objective of the first experiment is to show differences within and between novices and experts in problem formulation. Hypotheses H1_a and H1_b specifically state differences between novices and experts. 2-sample *t*-test can be used to test differences of means of problem formulation characteristics. Descriptive statistics can show other differences among participants and possibly lead to proposing new hypotheses. Unsupervised data mining methods such as sequence mining can reveal patterns within participants which may lead to generating other hypotheses.

The second experiment is about understanding the relation between problem formulation and ideation. This provides the answer to the main research question and also facilitates testing hypotheses H2_a and H2_b. The experiment involves finding correlations between pairs of problem formulation characteristics and ideation metrics. It also includes building regression and classifier models with formulation characteristics as

the independent variables and ideation metrics as the dependent variables. Testing hypothesis H2_b requires examining whether creativity is improving along a timeline which involves formulating several problems. Test of differences in means of ideation metrics for those problems facilitates testing hypothesis H2_b.

The third experiment examines if creativity can be predicted from problem formulation, testing hypothesis H3. The results of the second experiment provide models of ideation metrics with respect to problem formulation characteristics. The models built based on one problem can be used to predict the creativity metrics for another problem. The differences between actual and predicted scores can be examined with paired *t*-test. The differences are expected to be zero. This can be tested with certain degree of confidence. The details of testing the hypotheses and whether they are proven or rejected will be described for each experiment in the following three chapters. The design of the experiment is summarized in Table 6.7. It should be stated that although it would be preferable to give different groups of designers the same design problems (block the design of the experiment against the “design problem” factor and considering it as a noise variable), especially in comparing experts and novices, this was not possible because of familiarity of some participating designers with design problems. The DP_1 problem assigned to the experts (F11E) was in a design textbook and given as a project to some of the participants in F12U. Additionally, there was some material about DP_1 available online when initial results of this research was published. Familiarity of the participants with the design task would have been a far more serious flaw in the design of the experiments compared to differences in problems. Finding problems that are similar with

respect to the characteristics discussed in section 6.1 is one of the challenges in this study which is also explained in section 11.2.

Table 6.7 The design of experiments

	Experiment I	Experiment II	Experiment III
Objective	Showing differences within and between experts and novices	Understanding the relation between problem formulation and creativity	Predicting creativity from problem formulation
Input	Problem formulation characteristics	Problem formulation characteristics Ideation metrics	Formulation-ideation models Ideation metrics
Collected data	Protocol P-maps Testbed P-maps	DT test scores Protocol P-maps Testbed P-maps Sketches	Testbed P-maps Paper sketches
Design problems	DP_1, DP_2, DP_3, DP_4, DP_5	DP_1, DP_3, DP_4, DP_5	DP_4, DP_5
Participants	F11E, F12U, F13G, F14G	F11E, F14G	F14G
Analysis methods	Descriptive statistics Test of differences (2-sample <i>t</i> -test) Unsupervised data mining	Correlation analysis Regression analysis Supervised data mining Test of differences	Regression analysis Test of differences (Paired <i>t</i> -test) Descriptive statistics
Output	Differences in formulation characteristics	Models of ideation vs. formulation Differences in ideation metrics	Differences between predicted and actual ideation
Hypotheses tested	H1_a, H1_b	H2_a, H2_b	H3

CHAPTER 7

EXPERIMENT I: DIFFERENCES IN EXPERTS AND NOVICES

One of the fundamental research questions in this study is to understand how different designers formulate problems. Several characteristics might differentiate designers from each other. One characteristic is the level of expertise. Many studies of designer thinking are about differences between experts and novices [2]. Learning these differences can lead to recommendations for successful designing. The objective of the first experiment is to understand how experts formulate problems differently from novices. To know if such differences are due to level of expertise, it is useful to learn if differences in problem formulation occur also within each of the expert or novice groups. In addition, problem formulation is an understudied subject and learning about how any of expert or novice groups perform adds to our knowledge of the phenomenon. Therefore, in addition to testing hypotheses, additional findings in this experiment can be considered a part of an exploratory study. Representing differences within each of the expert and novice groups also leads to such findings. Observations will be reported about trends in each group and significant differences between groups, but two specific hypotheses will also be tested. These two hypotheses are based on an earlier exploratory study [77] which was explained in section 3.1. They are:

H1_a) Novices follow a systematic order in expressing problem formulation while experts have a more opportunistic behavior.

H1_b) Experts find key issues early on during problem formulation while novices find more issues and later in the formulation process.

Trends in problem formulation characteristics will also be shown within novices. If there is a positive correlation between a problem formulation characteristic and a creativity measure in different problems, and if there is a positive trend in both the problem formulation characteristic and the creativity measure, then it can be inferred that problem formulation can be improved with practice in novices. Finding the relation between problem formulation characteristics and creativity is done in Experiment II. Therefore, the trends found in this experiment will be used in examining if creativity can be improved among novices with practice (H2_b in Experiment II). In order to achieve the objectives of Experiment I, data was collected from experts and novices, problem formulation characteristic were extracted, and differences between and within the two groups were represented.

7.1 Collected data

To find the differences between and within experts and novices problem formulation data was collected from four groups of participants. The first group, F11E, consisted of eight expert designers in a consumer electronics company. They were asked to think aloud while they worked on the water sampler problem (DP_1) in an hour-long session. They were videotaped and their notes and sketches were also collected. The second group of participants, F12U, was about sixty undergrad students. They were asked to work on the can crusher problem (DP_2) in the Problem Formulator testbed. The third and the fourth group of participants were mechanical engineering graduate students (F13G and F14G). The F13G group worked on the goofy gopher (DP_3) and the autonomous

surveillance vehicle (DP_5) in the Formulator. The F14G group worked on DP_3 and DP_5 in addition to the shot buddy problem (DP_4) in the Formulator.

For this experiment, problem formulation characteristics were analyzed. While the data collected from the students in the Formulator was readily described in the P-maps ontology, the protocols collected from the experts had to be encoded into P-maps. Transcription, segmentation, and coding of protocols into P-maps were carried out similarly to the protocol analysis process described in sections 3.1 and 3.2 with one difference. There was a predefined coding schema: the P-maps ontology. While the protocol analysis method described in Chapter 3 led to the development of P-maps ontology, protocol analysis for this experiment led to coded data within the P-maps ontological framework. Using a predefined coding schema is not common when using protocol analysis, but it is not unprecedented; an example is Pourmohamadi and Gero [50] who used F-B-S [48] as a coding schema. Protocols of the eight experts were coded into P-maps through a process of arbitration between two expert researchers. A sample protocol with coding is given in Appendix C.

The problem formulation characteristics which were analyzed in this experiment were some state counts and a few strategies. Three of the state counts were considered:

- The counts of each entity (e.g., total number of requirements).
- Percentages of entities (e.g., total number of issues divided by total number of all entities).
- Median occurrence of entities (e.g., the relative position where half of the issues were added if the position of the first and last entities were considered 0 and 1 respectively).

Besides the state counts, occurrences of three strategies were traced and counted. They were abstraction, forward order, and entity depth prevalence. It should be noted that the data collected from groups F11E and F12U was based on an earlier version of the ontology which did not have the *Use scenarios* entity. Therefore, all strategies and all state counts were not considered in examining the differences between and within experts and novices. It was still possible to use the problem formulation characteristics which were considered to test the hypotheses stated for this experiment. Some of the problem formulation characteristics which were chosen specifically relate to testing hypotheses H1_a and H1_b. The forward order strategy is used in testing H1_a. The percentage of issues and median occurrence of issues are used in testing H1_b.

7.2 Analysis method

To show the differences within each group of participants and between experts and novices, the collected data was analyzed in two ways. One was to use simple data visualization and descriptive statistics. Three types of plots were mostly used in describing differences among designers. They are time series plot, run chart (also known as sequence plot), and Boxplot. Time series plots are mainly used to show how many entities of different types designers add during formulating a problem. Sequence plots or run charts are similar to time series plots with a slight difference. While in time series plots the Y axis is a numerical variable (e.g., count of added variables up to the time on the X axis), in the sequence plot the Y axis is a set of nominal variables (e.g., name of the entity types). Sequence plots can show the duration of attention paid to a specific entity and the frequency of shift in attention to different entities. Boxplots are another

descriptive statistics tool to show differences among designers or groups of designers. Boxplots provide a compact representation of the tendency (median) and the dispersion (range and interquartile range) in the data.

The other method of analysis that was used was test of means with 2-sample t -test. In order to understand if the differences between two groups of designers are statistically significant, a hypothesis test of differences between the means of specific variables should be done. The stated hypotheses described earlier should be translated into formal hypotheses of differences in means accordingly. For example, hypothesis H3 states that experts find issues earlier than novices and novices add more issues. This can be restated as the difference between two means for variables explained in the previous section: total number of issues, and median occurrence of issues. T -test is used for hypotheses test on the difference in means of two samples when their variances are unknown. There are two cases. When the unknown variances are considered equal, t -statistic with $n_1 + n_2 - 2$ degrees of freedom is used. When variances cannot be considered equal, an approximate t -statistic and degree of freedom are used:

$$T_0^* = \frac{\bar{X}_1 - \bar{X}_2 - \Delta_0}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}}, \text{ and } v = \frac{\left(\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}\right)^2}{\frac{\left(\frac{S_1^2}{n_1}\right)^2}{n_1 - 1} + \frac{\left(\frac{S_2^2}{n_2}\right)^2}{n_2 - 1}}$$

where \bar{X}_1 and \bar{X}_2 are sample means and S_1^2 and S_2^2 are sample variances. The null hypothesis that is tested assumes that the difference in the means is equal to an amount Δ_0 (which is often assumed zero), i.e., $H_0: \mu_1 - \mu_2 = \Delta_0$. One benefit of t -tests in test of differences of means is that they are often valid even when the populations moderately

deviate from normality [106]. Confidence intervals are also found using the T_0^* approximation. An approximate $100(1 - \alpha)\%$ confidence interval on the difference in means $\mu_1 - \mu_2$ is found from:

$$\bar{x}_1 - \bar{x}_2 \pm t_{\alpha/2, v} \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$$

In addition to testing specific hypotheses and searching for differences within of participants, search for similarities and patterns can lead to new observations and generating new hypotheses. Once a large number of P-maps are collected, data mining can be used to search for patterns. One method that was used with P-maps is sequence mining. P-maps can be written as a sequence of the entities, attributes, and links that a designer adds in the order of creation. The sequences can be searched for frequent subsequences with high measure of support; that is to see how frequently a partial order of the entities appeared among different designers [107]. Sequence mining could be used to test hypothesis H1_a which stated that novices follow a systematic order in expressing problem formulation. However, occurrences of strategies relating to specific orders were better characteristics for testing H1_a. Sequence mining among novices revealed another pattern.

7.3 Results and conclusions

7.3.1 Representing differences within experts

To demonstrate the differences in problem formulation within experts, the coded protocols of the eight expert designers (F11E) working on the water sampling problem (DP_1) were analyzed. The P-maps data model for this data set does not include *Use*

scenarios. The overall number of entities within the five entity types were plotted over a normalized timescale to eliminate differences in the length of the design sessions. Each coded predicate equaled one time step. Figure 7.1 shows the normalized time series plot for two experts. One expert specified more problem-related entities of the design by continuously adding new requirements and functions. Contrastingly, the other expert focused on solution-related entities, especially by specifying more behaviors. The designer that defined requirements throughout the design process was atypical and in fact, the other designers specified requirements towards the beginning of their sessions.

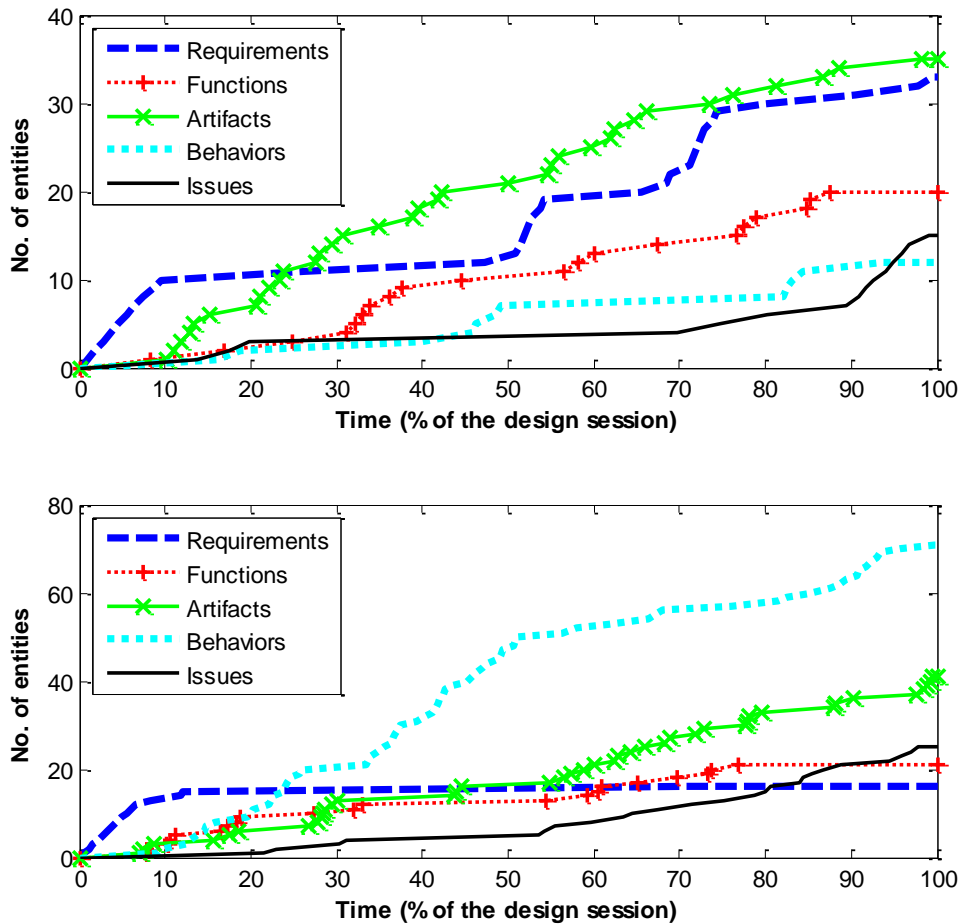


Figure 7.1 Time series plots of entities for two experts (from [108])

Although designers have different styles of problem solving that are not dependent on the solution [20], there are some similarities in the ways in which they move among the five groups of entities. To see whether or not the designers formulated the problem in a similar order, run charts were used. Figure 7.2 compares how two designers (different from those compared in Figure 7.1) moved among the five groups of entities. The iterations show that the process of defining artifacts, behaviors, and functions was strongly intertwined. However, one designer (the top graph) develops an entity type before moving to another entity type of the problem, while the other designer (the bottom graph) quickly shifts attention to different entities.

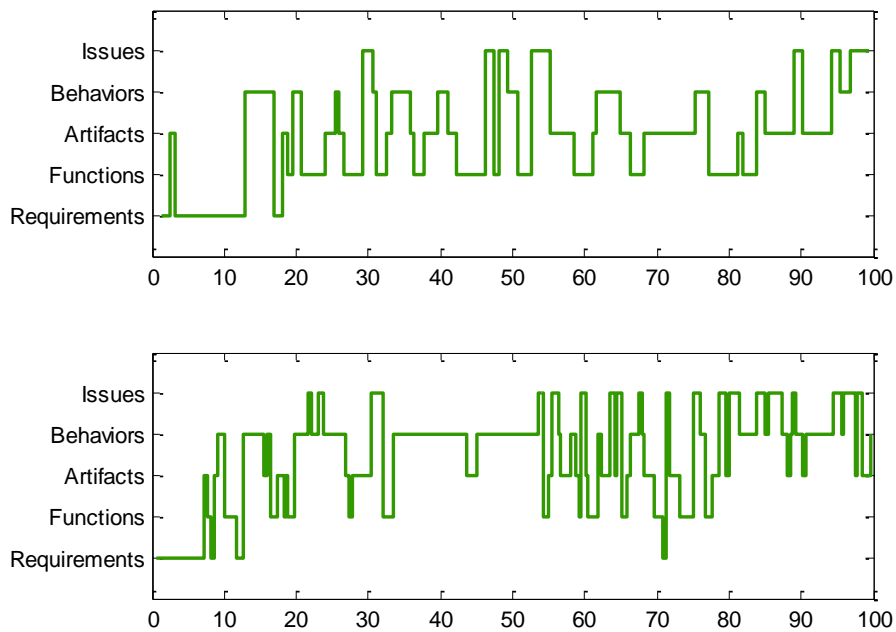


Figure 7.2 Comparison of iterations among entities for two experts (from [108])

In general, the eight experts often went back and forth quickly between defining their artifacts and their functions. For those who also spent substantial effort identifying

behaviors, the behaviors were often intertwined with functions and artifacts. Another interpretation of the drawn run charts is to characterize designers' attention with the duration of micro-level intervals of staying on one type of entity during problem formulation. While the top graph shows attention intervals of a relatively equal length throughout problem formulation, the bottom graph shows a change from long attention to an entity type to short attention spans to an entity. This suggests that in addition to characterizing designers with depth-first vs. breadth-first exploration of entities, it is possible that different combinations of both exploration strategies are present among designers.

7.3.2 Representing differences within novices

Data has been collected from three groups of novices; one group of undergrads and two groups of grad students. This provides an opportunity to look into difference within novices in more than one way. First, the two groups of grad students (F13G and F14G) were compared. Both groups had taken the same course and had worked on three design problems in similar situations. It was possible to look at changes in problem formulation characteristics along the course timeline and for the two years. The rates of occurrences of two problem formulation strategies were compared.

Figure 7.3 and Figure 7.4 show the changes in the number of times the students adopted the *abstraction* and *entity depth prevalence* strategies in the two groups for three design problems. It can be seen that in both groups, there was a rise in the adoption of the strategies throughout the course, even though there was a wider distribution among students of 2013.

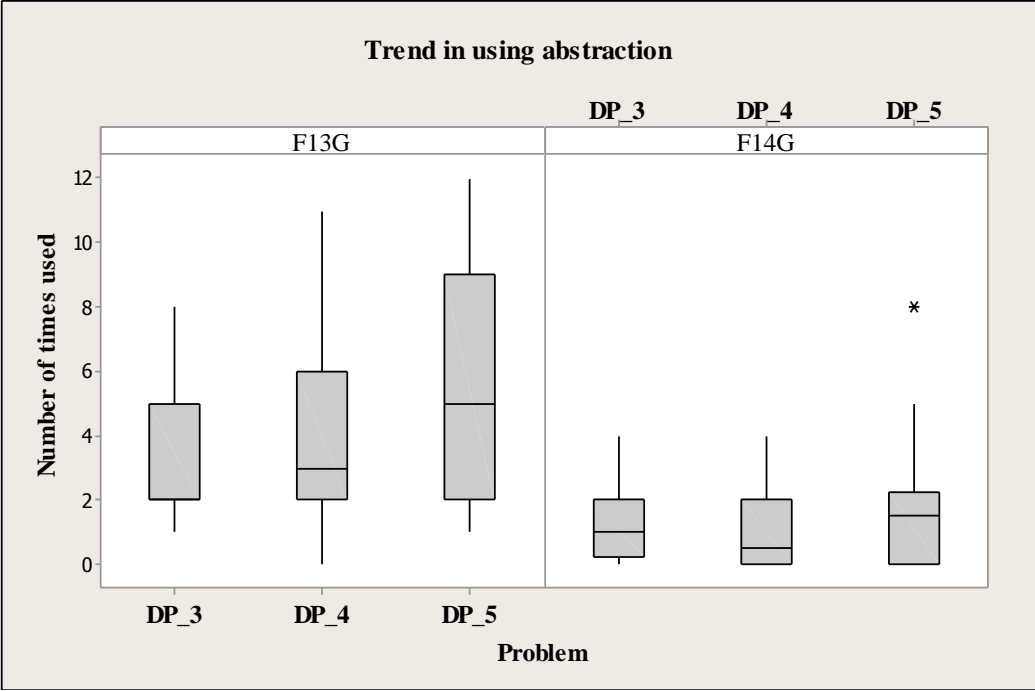


Figure 7.3 Comparing trends in using abstraction for two classes of students

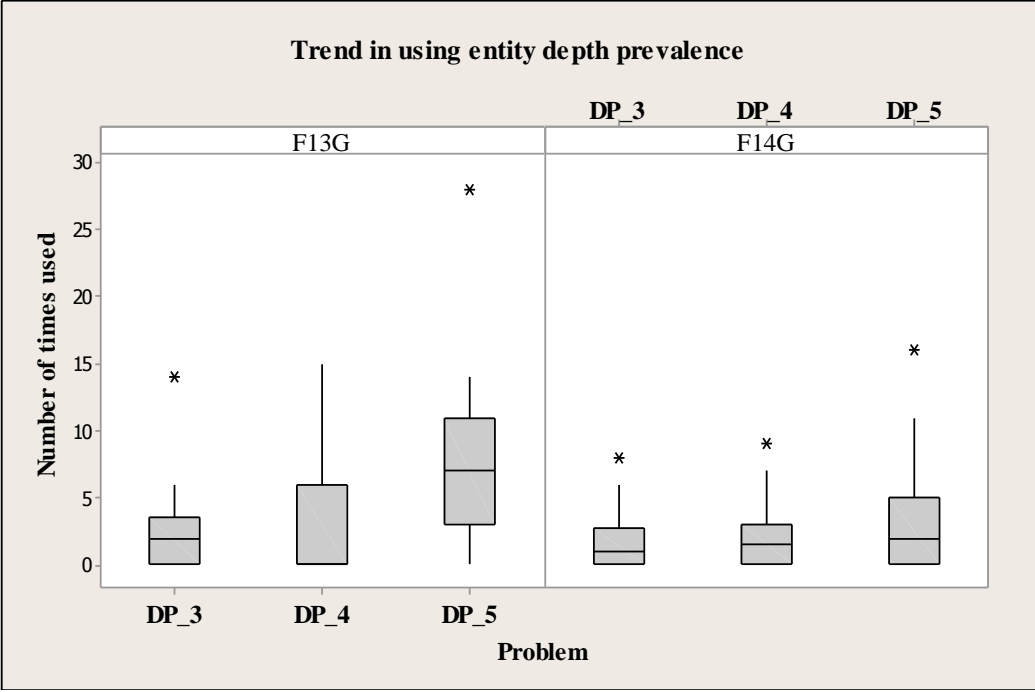


Figure 7.4 Trends in using entity-depth-prevalence for two groups of students

Examining the changes over time for one group of participants is another way of finding differences within novices. This is done specifically in relation to hypothesis H1_b. It states that experts find key issues early while novices find more issues and later in the formulation process. Comparison of novice and experts will be shown later in this section. Here, changes in percentage of issues (total number of issues divided by total number of entities) and the time of adding issues across multiple problems can be examined. The results of test of differences in means with two-sample *t*-test for one group of students (F14G) are shown in Table 7.1 and Table 7.2. Participants worked on problems DP_3, DP_4, and DP_5 in weeks 2, 6, and 10 of their design course.

Table 7.1 Change in novices' time of discovering issues through practice

	Difference (normed median)	95% CI	P-value
DP_3 – DP_4	-0.0246	(-0.1729, 0.1236)	0.734
DP_4 – DP_5	0.2008	(0.0600, 0.3415)	0.006

Table 7.2 Change in the % of issues novices discover through practice

	Difference %	95% CI	P-value
DP_3 – DP_4	-0.02335	(-0.04296, -0.00375)	0.021
DP_4 – DP_5	0.0003	(-0.0202, 0.0208)	0.978

Toward the end of a semester-long course, problem after problem, the students discovered issues earlier in formulating a design problem. The variable for which the difference of means is tested is the median occurrence of issues. It specifies (on an interval scale of 0-1) when half of the issues were added. The difference between DP_3 and DP_4 is insignificant but the median occurrences of issues shifts 0.2 of the duration

of the problem formulation to the beginning from DP_4 to DP_5 (median occurrence of DP_4 is larger than DP_5, i.e., issues are added later in DP_4). In other words, the students learn with practice to discover issues earlier. The students also learned to identify more issues. Table 7.2 shows the differences in means with two-sample *t*-test for the issues as a percentage of all entities. The results show that there was an increase of about 2% in issues as a percentage of all entities from DP_3 to DP_4. There were no significant changes between the last two problems.

In addition to showing differences within novices, search for patterns was conducted using sequence mining. The largest data set for which sequence mining was done belongs to the F12U group (about 60 undergrad students) working on the DP_2 problem. Figure 7.5 shows a sequence collected from one of the students. Table 7.3 shows results of the frequent sub-sequences with a support measure more than 0.5, indicating that they occurred among more than half of the students. Not surprisingly, the common patterns among the students are those of specifying a few requirements or functions in a row, following a requirement with a function, and developing a hierarchy of requirements and functions. None of the frequent sub-sequences have any of the other entities in the ontology or linking entities after adding them. This may suggest that students are problem-oriented rather than solution-oriented [34].

'requirement', 'function', 'requirement', 'parent_of_requirement', 'requirement',
 'requirement', 'requirement', 'requirement', 'parent_of_requirement', 'requirement',
 'parent_of_requirement', 'parent_of_requirement', 'requirement', 'requirement',
 'parent_of_requirement', 'requirement', 'parent_of_requirement', 'function',
 'function', 'function', 'parent_of_function', 'function', 'function', 'requirement',
 'requirement', 'parent_of_requirement', 'requirement', 'requirement',
 'parent_of_requirement', 'requirement', 'requirement', 'satisfies', 'satisfies'

Figure 7.5 An example of a P-maps sequence

Table 7.3 Frequent sub-sequences with a support higher than 50%

Sequence	Support
['requirement', 'function']	0.59
['function', 'function', 'function']	0.59
['requirement', 'requirement', 'requirement']	0.62
['requirement', 'parent_of_requirement', 'requirement']	0.51
['parent_of_requirement', 'requirement', 'requirement', 'parent_of_requirement', 'parent_of_requirement', 'requirement', 'requirement', 'parent_of_requirement']	0.54
['parent_of_function', 'function', 'function', 'parent_of_function', 'parent_of_function', 'function', 'function', 'parent_of_function']	0.57

7.3.3 Testing differences between experts and novices

The last part of Experiment II is to find differences between experts and novices. Similarly to the search for differences between the two grad student groups, the first comparison was made in the rate of adopting two strategies: *abstraction* and *forward order*. The results are shown in Table 7.4. They suggest two things. One is that experts

use more abstraction than novices do. The difference in the means of occurrences of the *abstraction* strategy are statistically significant ($T_0^* = 1.8416$, $v \cong 8$, $\mu_1 - \mu_2 = 3.2$, $p = 0.041$, and 95% CI [0.16,6.24]). The other is that novices are more likely to follow a specific order, which in this context means that the designer adds entities in an order from requirements to issues. The difference in the means of occurrences of the *forward order* strategy are statistically significant ($T_0^* = 2.8466$, $v \cong 67$, $\mu_1 - \mu_2 = 0.97$, $p = 0.006$, and 95% CI [0.29,1.65]).

Table 7.4 Variations in adopting two strategies among students and experts

	Students (n=62)		Experts (n=8)	
	<i>Abstraction</i>	<i>Forward order</i>	<i>Abstraction</i>	<i>Forward order</i>
Mean	2.9	1.1	6.1	0.13
Median	3	0	4.5	0
STD	2.7	2.5	3.6	0.35

To complete the test of hypothesis H1_b, the percentage of issues and the median occurrences of issues were compared between novices and experts. The results are shown in Table 7.5 and Table 7.6. The results suggest that the students discover fewer issues compared to the experts (about 3% on their first problem). They also suggest that there is no significant difference between students and experts in the time of discovering issues.

Table 7.5 Differences between experts and novices in the amount of issues

Novice (F14G) – Expert (F11E)	Difference %	95% CI	P-value
DP_3 – DP_1	-0.03612	(-0.05562, -0.01663)	0.001
DP_4 – DP_1	-0.01277	(-0.03139, 0.00585)	0.169
DP_5 – DP_1	-0.01305	(-0.03337, 0.00726)	0.199

Table 7.6 Differences between experts and novices in the time of adding issues

Novice (F14G) – Expert (F11E)	Difference	95% CI	P-value
DP_3 – DP_1	0.0541	(-0.1004, 0.2086)	0.470
DP_4 – DP_1	0.0788	(-0.0543, 0.2118)	0.230
DP_5 – DP_1	-0.1220	(-0.2689, 0.0249)	0.100

To summarize, the following conclusions can be made based on the inferences from the collected data for Experiment I:

- Hypothesis H1_a stated that novice designers follow a systematic order in expressing problem formulation while experts have a more opportunistic behavior. The results of comparing the rate of adoption of the *forward order* strategy (Table 7.4) showed that novices were more likely to follow adding entities in the specific order from requirements to functions, artifacts, behaviors, and issues. Hypothesis H1_a is therefore proven.
- Hypothesis H1_b stated that experts find key issues early on during problem formulation while novices find more issues and later. Results of the test of differences of means (Table 7.5 and Table 7.6) showed that novices discovered fewer issues compared to experts, but there was no significant difference between novices and experts in the time of discovering issues. Hypothesis H1_b is therefore rejected.

In addition to testing hypotheses H1_a and H1_b, Experiment I also led to the following findings:

- Experts use abstraction more than novices do (see Table 7.4).

- Based on novices' frequent sub-sequences (in Table 7.3), a new hypothesis can be proposed which states that novices are problem oriented.
- The experts' run charts (Figure 7.2) suggest that at the micro-level, designers' span of attention changes during problem formulation. While some designers have a relatively constant attention span for each entity type, others may have changing attention spans during problem formulation (e.g., from long focus on an entity to quick shifts of attention across entities).

CHAPTER 8

EXPERIMENT II: RELATING FORMULATION TO CREATIVITY

The central hypothesis of this thesis is that problem formulation influences creativity. To support this hypothesis, the relation between problem formulation and creativity should be understood. The objective of the second experiment is to model this relation. To build models of creative problem formulation, characteristics of problem formulation and creativity should be defined. P-maps variables characterize problem formulation. Divergent Thinking test scores [99] and ideation metrics [24] are apriori and aposteriori characteristics of creativity. Understanding the relation between problem formulation and creativity is the key to examining two of the stated hypotheses:

H2_a) Depth-first exploration of problem formulation entities leads to more creativity.

H2_b) Creativity can be improved in novice designers by teaching them characteristics of good problem formulation.

Examining hypothesis H2_a is about determining whether there is a significant correlation between a specific strategy and creativity measures. Examining hypothesis H2_b requires measuring the change in creativity across several problems. In the previous experiment, there were observations of changes in problem formulation characteristics. Progress in problem formulation that leads to improved creativity can be evaluated with test of differences in means of ideation metrics across multiple problems for the group as a whole (2-sample *t*-test), or for individuals (paired *t*-test). P-maps have several characteristics of problem formulation. There are opportunities in finding other

significant relations in models of problem formulation characteristics with respect to creativity measures.

8.1 Collected data

To understand the relationship between problem formulation and creativity and progress in a course, two data sets were investigated. The first was the P-maps from encoded protocols of the eight experts of F11E group working on problem DP_1, in addition to their Divergent Thinking test scores. The second data set was the Formulator testbed P-maps and concept sketches collected from the graduate students of F14G group. They worked on problems DP_3, DP_4, and DP_5. Different problems are used for a few reasons. One is to collect more data and have a larger sample. The other is to see if there are trends that are common across different problems. Some analyses e.g., correlation are less sensitive to the magnitude of the variables. When two sets of variables (a problem formulation characteristics and an ideation metric) are examined for correlation, the magnitude of scale for each variable does not affect the coefficient. In fact, the Pearson correlation coefficient is invariant to separate linear transformations in each variable. However, it should be noted that problem formulation characteristics in one problem are not a linear transformation of the same characteristics in another problem. This is a limitation in using more than one problem which will be discussed further in section 11.2. Yet, insensitivity of the correlation analysis to variable scale, and discretization for building classifiers are two remedies in dealing with this limitation. The problem formulation characteristics which were analyzed in this experiment were:

- The counts of each entity (e.g., total number of requirements).

- The counts of each isolated entity (e.g., total number of functions which are not in a hierarchy, i.e., they are neither a parent nor a child node).
- The counts of each disconnected entity (e.g., total number of requirements which are not linked to any other entity type).
- The counts of occurrences of all strategies except *forward order* (it had no occurrences for any of the problems).

Due to limited availability of the experts, they were not asked to continue their design process from problem formulation to ideation. Therefore, ideation metrics could not be used for them. Instead, their Divergent Thinking test scores were used as an apriori measure of creativity. For the students, concept sketches were collected for each problem and ideation metrics (quantity, variety, novelty, and quality) were found; the process was described in 6.4. For some of the analyses, the best novel idea and the idea with the highest quality were also considered. In those analyses, there are separate labels for average and max novelty and quality. This takes into account that some designers might generate many mediocre or good ideas but some designers come up with a few novel ideas.

8.2 Analysis method

To model the relation between formulation and creativity four analysis methods are applied. They are correlation analysis, multiple linear regression, decision trees, and test of differences of means. Correlation analysis is an extension of linear regression except that both variables of interest are jointly distributed random variables [106]. To determine the significance of a correlation the appropriate statistic is:

$$T_0 = \frac{R \sqrt{n-2}}{\sqrt{1-R^2}}$$

where R is the correlation coefficient (square root of the coefficient of determination which is found from the ratio of the sum of squares of regression SS_R to total sum of squares SS_T). T_0 has a t distribution with $n - 2$ degrees of freedom. Based on the value of the t distribution for a desired α , values for statistically significant correlation coefficients can be found.

While correlation analysis determines the relation between two variables, a multiple linear regression model finds the relation between multiple independent variables and a dependent variable. The general equation for a linear regression model with k independent regressors is:

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k + \epsilon$$

where β_j is the regression coefficient, which is the expected change in response Y per unit change in x_j when other regressors are held constant [106]. Regression coefficients are estimated with the least square method. A few statistics are used to test the significance of the model. One is to conduct analysis of variance. The F statistic is used. For a model built from n observations

$$F_0 = \frac{MS_R}{MS_E} = \frac{SS_R/k}{SS_E/(n-p)}$$

where MS_R and MS_E are mean square of the regression and the residual error, and $SS_E = SS_T - SS_R$ ($p = k + 1$, since β_0 is the constant). The mean sums of square are chi-square random variables, thus the regression is considered statistically significant

when F_0 is larger than $f_{\alpha,k,n-p}$. Two other metrics for determining the significance of the model are the coefficient of determination R^2 and the adjusted coefficient R_{adj}^2 . R^2 can be misleading as it is inflated with a large of number of regressors in the model. R_{adj}^2 takes the number of regressors into account since:

$$R_{adj}^2 = 1 - \frac{SS_E / (n - p)}{SS_T / (n - 1)}$$

In addition to testing the significance of the model, each regressor can be tested individually. The t statistic is used and when $|T_0| > t_{\alpha/2, n-p}$ the regressor is statistically significant at the specified α level.

The correlation analysis and multiple linear regression find the relation between numerical variables (nominal variables can be included in a multiple linear regression model as independent variables, but they are not discussed here). To build a classifier (a model for nominal dependent variables), supervised data mining methods should be used. The reason why a classifier is used in parallel with linear regression is to mitigate sensitivity to scales for both the problem formulation characteristics and the ideation metrics. Amabile [5] states that tests of creativity with a numerical score are sensitivity to differences among individuals. Though the ideation metrics are aposteriori measures of creativity, not tests, it may be helpful to suspend the assumption that creativity can be measured on a continuous numerical scale. In addition, the problem formulation characteristics that are defined so far are not established as metrics and might be sensitive to scale. Therefore, differences in individuals with respect to each formulation characteristic should be looked at in comparison to the other participants in the sample.

Instead of considering raw counts, the participant's measure of a characteristic is reported e.g., as low, medium, and high in the sample. Another reason why it was more appropriate to turn some problem formulation characteristics into nominal variables was to reduce sparseness in the space of the variables which had an uneven distribution. Most of the problem formulation strategies had zero occurrence in half of the participants while there were also a wide range of occurrences among others. Finally, turning the quantities into nominal variables also makes it possible to combine data from different problems and also lessen the effect of the design problem.

The ideation metrics are discretized into nominal variables to build classifiers with problem formulation characteristics as the attributes. Though the correlation and linear regression models provide a mathematical equation of the studied relations, they are built on the assumption that the variables are continuous and numerical. With classifier models, it is likely that patterns are found in problem formulation that lead to being more creative or less creative (or have high, medium, and low creative outcome). To build classifiers of creativity with respect to problem formulation characteristics decision trees are used. It is because they are easy to construct, easy to interpret (for small-sized trees), and they are accurate compared to other classification methods [107].

To evaluate the performance of decision trees for comparing different models several metrics can be used. A simple metric is accuracy which is the percentage of correctly classified instances. This measure is more appropriate when instances per class are not too different (e.g., there are as many instances in the data labeled more creative as labeled less creative). Two other measures that take into account the balance in the number of instances per class are precision and recall. Precision is the number of true positives (i.e.,

the number of instances correctly labeled as belonging to the positive class) divided by the sum of true positives and false positives (i.e., the number of instances incorrectly labeled as belonging to the positive class). Recall is the number of true positives divided by the sum of true positives and false negatives (i.e., the number of instances incorrectly labeled as belonging to the negative class). While precision shows how many classified instances are correctly assigned to a label in the class, recall shows how many instances that belong to a label are found. The three metrics described evaluate the performance of a model. To compare the performance of competing models when building a classifier, the appropriate metric is the Receiver Operator Characteristic (ROC) curve. ROC curves plot true positives (TP) against false positives (FP). The ideal case is that all models correctly classify all instances, i.e., $TP=1$ and $FP=0$. If the TP-FP values for the classifiers lie on the line between $TP=0$ and $FP=0$ (every instance classified as the negative class) and $TP=1$ and $FP=1$ (every instance classified as the positive class) the models are randomly guessing. The area under the ROC curve is 1 for the idea case and is 0.5 for random guesses.

There are several algorithms used for building decision trees, but here the common C4.5 algorithm [109] is used in the Weka data mining software [110]. The last method of analysis applied in Experiment II is the test of differences in means (two-sample t and paired t tests). This is specifically pertinent to testing hypothesis 2 which examines if creativity is improving after formulating several problems during a course.

8.3 Results and conclusions

8.3.1 Correlation analysis

Correlation analysis was conducted for two data sets in this research. The first one involved the experts' protocol data (F11E) to find the correlation between a few P-maps variables and the Divergent Thinking test [99] scores of the participants as an apriori measure of creativity. The results are shown in Table 8.1. For eight participants, significant correlations should be above 0.62 with $p < 0.1$ (italic text), and 0.71 with $p < 0.05$ (bold text) respectively. The following P-map variables were measured:

- Total number of overall entities.
- Total number of links between entities.
- Average number of vertices; a measure of connectedness of entities.
- Total number of each entity (requirements, functions, artifacts, behaviors, and issues)
- Total number of parent-child relationships; the intra-group relationship specifying hierarchical information in the P-maps.

The results suggest that an overall increase in the total number of expressed entities is more likely to have occurred among designers with better divergent thinking skills. More specific correlations were also present. Number of specified behavior entities strongly correlated with the overall creativity level of the designers, as well as the ability to generate more ideas (fluency) and the ability to have concepts with higher quality. This can be expressed by having better domain knowledge since behaviors are expressions of technical knowledge. Quality of ideas also was correlated to more number of issues

identified. Quality of an idea relates to its feasibility and identifying design issues is a part of feasibility analysis. The elaboration in building a hierarchical structure (the number of parent-child relations) correlated with the designer's ability to come up with novel ideas (originality and max originality). This might be explained by Koestler's bisociation theory in creativity [111] which states creativity arises from combinations in structured thought.

It should be mentioned that there were a few correlations within each sets of variables. Within P-maps characteristics, besides an expected correlation between the number of entities and links (0.93, $p < 0.001$), there is a significant correlation between the number of functions and artifacts (0.95, $p < 0.000$), and issues and behaviors (0.68, $p < 0.066$). Within DT test scores, there is a correlation between fluency and flexibility (0.65, $p < 0.083$), originality and abstractability (-0.7, $p < 0.052$), and quality and abstractability (0.73, $p < 0.041$). Although the correlation between fluency and flexibility has been reported in the development of the DT test [99], the correlations which abstractability has to originality and quality are unusual. This may relate to the small sample size and similar background of the experts who were not randomly chosen from a larger pool.

In addition, decomplexability was inversely correlated with the average number of vertices. This may point to more creative designers expanding the design space in one direction rather than thinking of many alternatives or decompositions, though further investigation is needed. Finally, the total number of functions specified during the session was inversely correlated with abstractability among the expert participants. These correlations did not provide any definitive answers regarding the role of creativity in

problem formulation, though they inspired a novel way of exploring this interaction among more participants.

Table 8.1 Correlations between DT test and P-maps for experts (from [108])

Divergent thinking test score	# of entities	# of links	Avg. vertices	# of parent-child	Total req.	Total fun	Total art.	Total beh.	Total issue
Overall score	0.63	0.6	0.06	0.46	-0.31	0.16	0.25	0.87	0.68
Fluency	0.54	0.4	-0.3	0.29	0.14	0.16	0.17	0.71	0.4
Flexibility	0.49	0.43	-0.01	0.4	-0.08	0.4	0.36	0.39	0.17
Avg. originality	0.44	0.43	0.1	0.77	-0.53	0.7	0.69	-0.02	0.01
Max originality	0.62	0.68	0.29	0.72	-0.62	0.62	0.61	0.42	0.36
Quality	0.26	0.36	0.26	0.04	-0.32	-0.34	-0.17	0.82	0.78
Decomplexability	-0.12	-0.45	-0.88	0.07	0.56	-0.26	-0.11	-0.11	-0.17
Detailability	0.13	-0.12	-0.59	0.33	0.44	0.44	0.36	-0.37	-0.51
Abstractability	-0.14	-0.1	0.06	-0.49	0.05	-0.72	-0.61	0.59	0.53
Afixability	0.19	0.01	-0.5	0.37	0.14	0.55	0.49	-0.33	-0.53

The second correlation study was conducted for the participants in the F14G group for the shot buddy (DP_4) and autonomous surveillance vehicle (DP_5) problems. For the twenty five participants in this group, correlation coefficients of magnitude 0.34 (less than -0.34 and more than 0.34) were statistically significant, with 95% confidence. The significant correlations between problem formulation characteristics and ideation metrics for the DP_4 and DP_5 problems are shown in Table 8.2. Similar correlations in both problems are in bold.

The results for problem DP_4 show a positive correlation between quantity and the number of raised issues, isolated artifacts, and isolated issues. One can infer that leaving

artifacts and issues in a flat list, i.e., not focusing on the architecture of the final product or organizing the issues lead to generating more ideas. There is also a high correlation between all five strategies except for entity depth prevalence and quantity; breadth expansion breeds quantity.

Table 8.2 Significant formulation-ideation correlations for students

Ideation metric	DP_4	DP_5
Quantity	Issues 0.45 Isolated issues 0.45 Order req_use 0.67 Order req_fun 0.67 Isolated artifacts 0.34 Abstraction 0.40 Conflict identification 0.43	Issues 0.36 Isolated issues 0.36 Order req_use 0.64 Order req_fun 0.57 Function 0.38 Disconnected artifact -0.40 Entity depth prevalence 0.53
Variety	Issues 0.40 Isolated issues 0.40 Abstraction 0.40 Conflict identification 0.42	Issues 0.43 Isolated issues 0.43 Order req_use 0.35
Avg. novelty	Isolated use scenarios -0.35 Entity depth prevalence 0.42	Isolated use scenarios -0.37 Disconnected function -0.41 Conflict identification 0.42
Max novelty	Isolated use scenario -0.35 Entity depth prevalence 0.40	Disconnected function -0.38 Disconnected artifact -0.37
Avg. quality	Disconnected issues -0.35	Behavior 0.38 Isolated use scenario -0.35
Max quality	Disconnected requirements -0.48 Conflict identification 0.40	

Having more issues in a flat list has a moderate positive correlation with variety as well. In addition, the more abstraction and conflict identification happened, the more likely it was for the students to come up with different types of concept solutions. Correlation results for both average and max novelty show that the more the use scenarios were left unorganized, the less the possibility of having original ideas. Additionally, higher rates of entity depth expansion led to more novel ideas; in other words, the more the students developed an entity before searching for (or being reminded

of) related entities in other categories, the more likely it was to propose novel solutions. Finally, students came up with solutions of higher quality when they did consider the relations between issues and other entity types. Best quality of solutions occurred when students did not fail in recognizing the relations between elicited requirements and other entity types, and when they identified conflicting requirements.

The results for the DP_5 problem show a few different significant correlations. The total number of identified functions has a moderate positive correlation with variety. The degree to which students made abstractions and found conflicts also have substantial correlations to variety too. An interesting difference between the correlations for the two problems is that entity depth prevalence is positively correlated with average and max novelty. As it will be discussed later, the progression of class over time, and the more constrained nature of the second problem resulted in an overall lower variability in the novelty of the students. It is plausible to infer that a more constrained problem requires more focus on each category of entities prior to the designer's shifting attention towards a different category, i.e., within-group depth exploration breeds novelty in more constrained problems. However, this does not contradict with the observation that the more the students failed in organizing the entities within each category and recognizing the relations to entities in other categories, the worse their ideas were in terms of novelty and quality.

8.3.2 Regression analysis

To have an understanding of how different variables in the problem formulation influence ideation metrics together, linear regression analysis was conducted for the data

set collected from the participants in F14G. Models were built for two problems: DP_4 and DP_5. First, a model was built with P-map state variables as the input variables, and each of the corresponding ideation metrics as the output. Separately, a model was built for the number of times different strategies were utilized during problem formulation with respect to the ideation results. The complete table of regressors for the state counts models can be found in Appendix D. Table 8.3 shows the coefficients of regression for the counts of occurrences of strategies (the column with the ‘Cons’ label shows the constant or the intercept). Significant regressors are shown in bold.

Table 8.3 Regressors of P-maps strategies counts models for two problems

Variable	Const	Abstraction	Entity depth prevalence	Order req_use	Order req_fun	Conflict identification
DP_4 quantity	2.31	0.73	0.18	3.67	-1.65	-0.82
DP_5 quantity	3.38	0.04	0.14	-0.2	1.46*	-2.06
DP_4 variety	3.03	0.8	0.31	4.66	-2.42	-1.02
DP_5 variety	5.7	-0.09	0.1	-1.67	1.70*	-1.23
DP_4 avg. novelty	4.42	-0.02	0.38	1.06	-0.67	-0.74
DP_5 avg. novelty	3.47	-0.02	0.1	-0.81	0.48	-2.29
DP_4 max novelty	5.62	0.26	0.47	1.77	-1.05	-0.94
DP_5 max novelty	5.04	-0.06	0.14	-0.87	0.23	-2.8
DP_4 avg. quality	5.02	-0.05	0.05	0.49*	-0.25	0.04
DP_5 avg. quality	4.23	0.11*	0.01	-0.26	0.55	0.25
DP_4 max quality	6.08	0.14	0.12*	0.62	-0.31	-0.23
DP_5 max quality	5.87	0.09	-0.04	0.4	-0.23	1.15*

Since regression analysis with this level of detail was unprecedented in a design thinking study, the criterion for choosing significant regressors was set not to be too strict. For P-map state counts, a p value below 0.2 was considered significant; for P-map

strategies counts, the bound was set at 0.1. If there were no p values below the set limit, the lowest p value was considered significant (those regressors are starred in the tables). Additionally, regressors that have the same sign in the models for the two problems are italicized. This comparison shows if both problems provide models that can have the same sense with respect to some variables, i.e., if some parts of the models are generalizable and insensitive to the problem. Among the P-map state counts models, average quality has the highest number of variables with similar signs for DP_4 and DP_5 (13) while max quality and variety have 4 and 5 variables with the same sign. One might infer that average quality is easier to predict for new problems.

In order to inspect how reliable the results were for the regression models, the coefficient of determination R^2 was used to check model fit. Table 8.4 shows the R-squared values for each of the regression models which were derived for the six corresponding ideation metrics. The test of significance of the model fit suggested that the P-maps state count model was more reliable than the strategies counts model. The results also suggested that average novelty and max quality had more reliable models in both problems.

Table 8.4 Test of model fit with R^2

Predicted variable	State counts		Strategies counts	
	DP_4	DP_5	DP_4	DP_5
Quantity	65%	75%	64%	33%
Variety	56%	57%	32%	8%
Avg. novelty	78%	65%	30%	25%
Max novelty	66%	72%	24%	35%
Avg. quality	72%	62%	7%	9%
Max quality	87%	71%	19%	7%

8.3.3 Improving model fit with backward elimination

The fit of the regression models is affected by the number of predictor variables in the model. Since all the variables in a linear regression model often do not significantly contribute to the variations in the dependent variable, the excessive independent variables should be removed from the regression model. It was described earlier how the adjusted coefficient of determination (R_{adj}^2) takes into account the number of variables in the model. R-squared is an inflated measure, i.e., the more the variables in the model, the higher R-squared is, even if most variables are not significantly contributing to the variations in the dependent variable. It can become misleading since fewer variables in the model lead to a drop in R-squared. Therefore, the R-squared adjusted statistic should also be considered for the large number of variables (compared to the number of data points) in the regression models. To improve model fit an iterative backward elimination process was adopted. The steps are as follows:

1. Build a regression model including all the input variables.
2. Find the regressor (input variable) with the highest p-value (least contribution to variability in the model).
3. Remove the least contributing regressor and build a new regression model.
4. Continue until R_{adj}^2 no longer increases.

The backward elimination process was carried out with the state and strategies variables combined to find a single model with a better model fit. The initial combined model had 23 variables (18 state counts and 5 strategies counts). The coefficients of the regression for the final models (with the highest R_{adj}^2) are listed in Appendix E. The

number of regressors common in models of both problems (DP_4 and DP_5) varies from 8 in the models of variety, and 13 in models of quantity and max quality. Models of variety have the least common regressors with the same sign (only 2), while models of average quality have 8 out of 12 regressors with the same sign for both problems with the same sign. For each ideation metric the average of the regressors that have the same sign in both problems can be used to create the model of creativity with respect to problem formulation characteristics. These models are as follows:

$$\begin{aligned}
 \text{quantity} = & 2.3 + 1.1 * \text{issue} - 0.9 * \text{isolated use scenario} + 0.1 \\
 & * \text{disconnected function} - 0.9 * \text{disconnected issue} + 0.8 \\
 & * \text{order req use} - 11.3 * \text{conflict identification}
 \end{aligned}$$

$$\text{variety} = 2.6 + 1.8 * \text{issue} - 0.3 * \text{disconnected requirement}$$

$$\begin{aligned}
 \text{novelty} = & 3.8 + 0.3 * \text{use scenario} - 0.4 * \text{disconnected use scenario} - 2.4 \\
 & * \text{order req use} + 1.8 * \text{order req fun} - 2.9 \\
 & * \text{conflict identification}
 \end{aligned}$$

$$\begin{aligned}
 \text{quality} = & 5.7 - 0.4 * \text{requirement} + 0.1 * \text{isolated requirement} + 0.4 \\
 & * \text{disconnected requirement} - 0.2 * \text{disconnected function} - 0.8 \\
 & * \text{disconnected issue} - 4.1 * \text{order req use} + 5.4 * \text{order req fun} \\
 & - 11.2 * \text{conflict identification}
 \end{aligned}$$

Though the number of variables with the same sign is lower compared to the initial state and strategies models shown earlier, the models are less complex (have fewer variables), have more statistically significant regressors, and have an improved model fit. The improved model fit after backward elimination can be seen in Table 8.5. The results show a large gain in R-squared adjusted in all models at no more than a 7% drop in R-squared (in the model of variety for DP_4). The smallest improvement occurs for the models of max quality which has an initial high predictability. The largest improvements occur for the models of variety and max novelty in DP_4. It should be noted that good model fit does not guarantee accurate prediction of new observations [106]. Measuring accuracy in predicting new observations is a part of Experiment III which will be discussed in the next chapter.

Table 8.5 Improvements in model fit after backward elimination

Predicted variable	DP_4				DP_5			
	Initial R^2	Final R^2	Initial R^2_{adj}	Final R^2_{adj}	Initial R^2	Final R^2	Initial R^2_{adj}	Final R^2_{adj}
Quantity	91%	88%	-9%	69%	98%	97%	70%	87%
Variety	75%	68%	-201%	31%	93%	92%	17%	72%
Avg. novelty	86%	87%	-37%	62%	90%	87%	-19%	70%
Max novelty	79%	72%	-150%	38%	88%	87%	-43%	54%
Avg. quality	96%	93%	48%	83%	84%	79%	-96%	30%
Max quality	100%	99%	95%	97%	88%	87%	-43%	60%

8.3.4 Classification with decision trees

Decision trees drew patterns of problem formulation characteristic attributes in relation to classes of ideation metrics. They were drawn for data collected from F14G (same data which used for multiple linear regression). To find general patterns, and

because of the small number of participants the data from the three problems were combined into one set.

The numerical ideation metrics were discretized into nominal class variables in two ways: equal width binning, equal frequency binning. With equal width binning, the range in the data was divided into three equal intervals; the bottom, middle, and top third were labeled low, medium, high creativity respectively. With equal frequency binning, the bins were set in such a way that the number of instances in each bin were nearly the same. Decision trees are not expressive enough for modeling continuous variables, therefore, the attributes (independent variables) were also discretized (equal width binning). The occurrences of the strategies for many the participants were zero; instead of partitioning the range into three equal width bins, they were coded into Yes or No. Discretization was done for data on each problem separately to control for the effect of the problem.

Two other options were considered in building the classifiers. One option related to testing the classifier. Two choices were decided upon: use the complete dataset as the training set; use five-fold cross validation. The classifiers with no test data are less reliable but more accurate. This increases the chance of finding patterns. Since the subject of this research is understudied the more patterns that are found have the benefit of generating new hypotheses.

The second option considered in building the classifiers was to choose the minimum number of instances per leaf node. The more the number of the instances per leaf nodes results in less forested (less complex) trees which are easier to interpret. The downside is having an under-fitting model and losing information. Similar to the justification for including classifiers with no test data, a higher number of instances per leaf node is

preferred. The classifiers were built with three choices of instances per leaf node: 3, 4, and 5. The resulting decision trees with the described options for the four ideation metrics are in Table 8.6 through Table 8.9. The two numbers in each leaf node are the number of correctly and incorrectly classified instances.

For each ideation metric, one of the trees was chosen based on the classification performance metrics described in the previous section. In data mining, different evaluation metrics are proposed for comparing different classifiers. However, there are no definitive rules for determining which classifier is superior, since often there is a trade-off among the tree performance criteria. The performance metrics of the chosen trees are italicized. For all ideation metrics, the models with no test set (training set only) are chosen since they have significantly higher accuracy. The low accuracy for the trees built with cross validation imply that they will not perform well for unseen instances (new observations) and are not reliable for testing new data. At this stage of the research the main priority is to build an accurate model of the existing observations that can also be interpreted. Therefore, another factor in determining which tree to choose is simplicity, i.e., having fewer leaves. In summary, selecting the tree was based on the following rules:

1. If the difference between the accuracies of two trees is less than 10%, choose the one with fewer leaves. If the difference is more than 10%, choose the one with more leaves as long as the tree is not twice as large.
2. If accuracies and number of leaves are close (5% and 3), choose the tree that has higher number of min instances per leaf nodes.
3. Disregard trees with fewer than 5 nodes and an accuracy less than 60%.

For trees with relatively similar number of leaves, the other metrics (accuracy, precision, and ROC area) are taken into account. The selected trees for the four ideation metrics are represented in Figure 8.1 through Figure 8.4. An interpretation of the quantity decision tree is that participants who did not link most of the functions had low or medium quantity. This implies that participants who only thought about what the design should do (function) without considering why it should be done (relation to requirements), how it functions when used (relation to use scenarios), and what possible solutions exist (relation to artifacts) to carry out the function did not come up with many ideas. If all functions were linked to other entities, participants who did not abstract had lower quantity. One participant with high a quantity score linked most of the functions (low disconnected function) and had all requirements hierarchically organized.

Table 8.6 Comparison of decision trees built for quantity

Class binning	Model characteristic	5 fold cross validation (left) vs. no test (right)					
		3 (min inst./leave)		4 (min inst./leave)		5 (min inst./leave)	
Equal width	Leaves	8	8	7	7	7	7
	Accuracy	45%	64%	47%	62%	48%	62%
	Precision	0.432	0.701	0.414	0.689	0.432	0.689
	Recall	0.452	0.644	0.438	0.616	0.479	0.616
	ROC area	0.521	0.746	0.538	0.721	0.562	0.721
Equal frequency	Leaves	24	24	6	6	6	6
	Accuracy	36%	73%	33%	55%	30%	55%
	Precision	0.362	0.752	0.326	0.563	0.293	0.563
	Recall	0.356	0.707	0.329	0.534	0.301	0.548
	ROC area	0.562	0.89	0.514	0.7	0.505	0.7

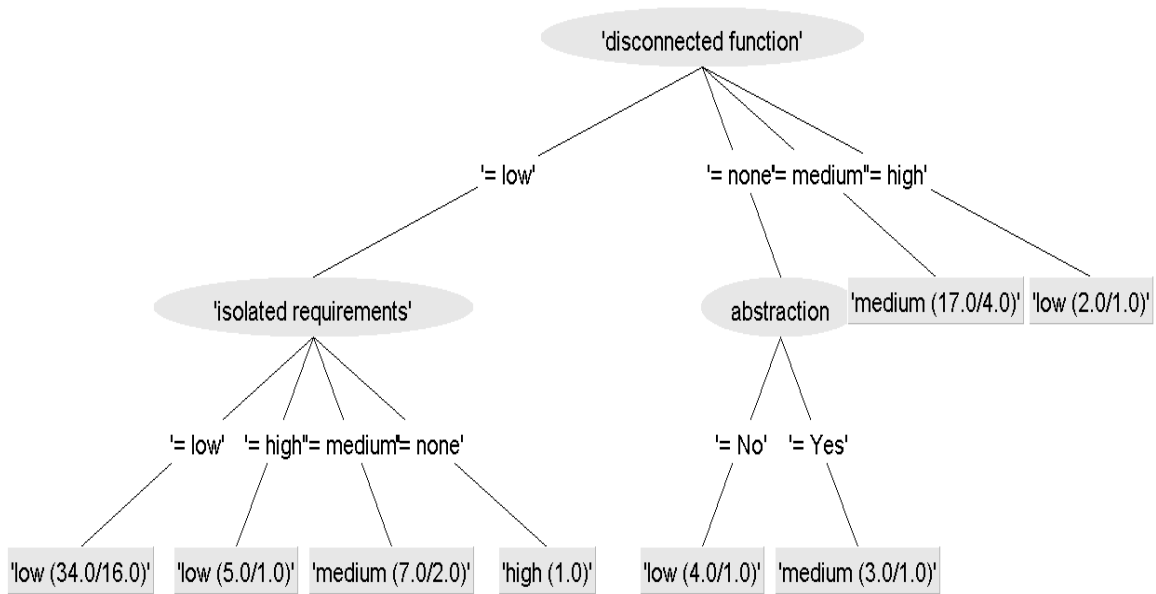


Figure 8.1 Selected decision tree for quantity

The variety decision tree suggests that designers who did not specify use scenarios or added too many of them were average in coming up with various ideas. Most variety of ideas occurred for participants with few use scenarios and requirements, or medium number of use scenarios but high number of issues. Having few issues led to low variety.

The selected novelty decision tree is more difficult to interpret since it is deeper and has more leaves compared to the quantity and variety trees. One observation is that participants who did not follow a breadth expansion order between requirements and use scenarios, had low number of functions and requirements, and had connected all or most of the issues had a high novelty. Having average number of functions and organized behaviors increases novelty. Having more functions with no breadth expansion between requirements and use scenarios also leads to high novelty.

Table 8.7 Comparison of decision trees built for variety

Class binning	Model characteristic	5 fold cross validation vs. training set					
		3 (min inst./leave)		4 (min inst./leave)		5 (min inst./leave)	
Equal width	Leaves	14	14	17	17	11	11
	Accuracy	32%	67%	36%	68%	38%	63%
	Precision	0.314	0.68	0.349	0.69	0.369	0.648
	Recall	0.315	0.671	0.356	0.685	0.384	0.63
	ROC area	0.444	0.812	0.456	0.844	0.467	0.783
Equal frequency	Leaves	18	18	18	18	9	9
	Accuracy	41%	74%	38%	74%	38%	63%
	Precision	0.423	0.768	0.39	0.757	0.388	0.634
	Recall	0.411	0.74	0.384	0.74	0.384	0.63
	ROC area	0.53	0.892	0.507	0.89	0.509	0.789

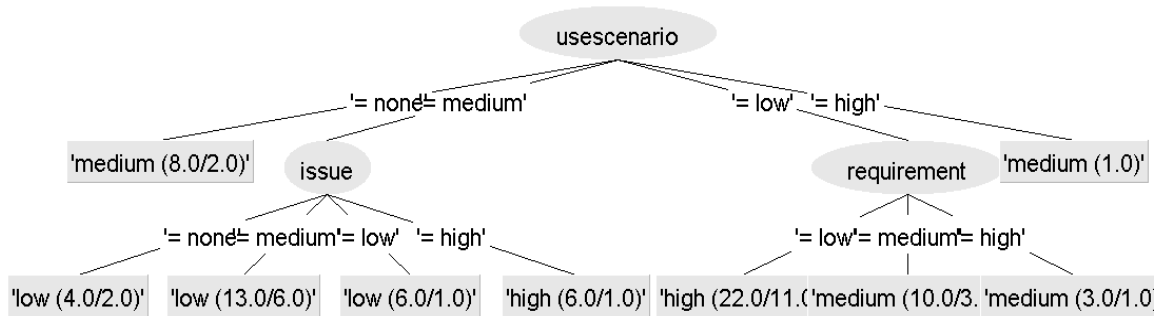


Figure 8.2 Selected decision tree for variety

In building the decision tree for quality, the low variability in the metric led to a disproportionate number of participants with a high score. Even when the width (range of scores) was divided into two bins there were still only 6 low instances compared to 67 participants with a high quality score. The classifier could not be built; Table 8.9 shows that for the equal width binning, all models have 1 leaf. Therefore, the tree was chosen

from after the data was discretized into three bins with equal instances per class. The selected tree can be seen in Figure 8.4. The tree structure suggests that participants who had few functions and did not expand entities in depth had a higher quality score. Having a medium number of functions but not many artifacts leads to low quality scores.

Table 8.8 Comparison of decision trees built for novelty

Class binning	Model characteristic	5 fold cross validation vs. training set					
		3 (min inst./leave)		4 (min inst./leave)		5 (min inst./leave)	
Equal width	Leaves	13	13	13	<i>13</i>	2	2
	Accuracy	58%	84%	56%	<i>84%</i>	53%	67%
	Precision	0.55	0.839	0.537	<i>0.839</i>	0.475	0.695
	Recall	0.575	0.836	0.562	<i>0.836</i>	0.534	0.671
	ROC area	0.508	0.894	0.52	<i>0.894</i>	0.454	0.585
Equal frequency	Leaves	18	18	9	9	5	5
	Accuracy	42%	77%	43%	60%	40%	52%
	Precision	0.439	0.808	0.433	0.662	0.394	0.581
	Recall	0.425	0.767	0.425	0.603	0.397	0.521
	ROC area	0.57	0.906	0.59	0.779	0.578	0.683

```

order_req_use = No
| function = low
| | requirement = low
| | | disconnected issue = none: high (10.0/1.0)
| | | disconnected issue = medium: medium (7.0/1.0)
| | | disconnected issue = low: high (9.0/2.0)
| | | disconnected issue = high: high (2.0/1.0)
| | requirement = medium: high (8.0)
| | requirement = high: medium (1.0)
| function = medium
| | isolated behavior = none: medium (5.0/2.0)
| | isolated behavior = medium: high (4.0/1.0)
| | isolated behavior = low: high (14.0/4.0)
| | isolated behavior = high: medium (4.0)
| function = high: high (4.0)
| function = none: medium (1.0)
order_req_use = Yes: medium (4.0)

```

Figure 8.3 Selected decision tree for novelty

Table 8.9 Comparison of decision trees built for quality

Class binning	Model characteristic	5 fold cross validation vs. training set					
		3 (min inst./leave)		4 (min inst./leave)		5 (min inst./leave)	
Equal width	Leaves	1	1	1	1	1	1
	Accuracy	-	-	-	-	-	-
	Precision	-	-	-	-	-	-
	Recall	-	-	-	-	-	-
	ROC area	-	-	-	-	-	-
Equal frequency	Leaves	17	17	11	11	11	<i>11</i>
	Accuracy	41%	78%	40%	68%	40%	<i>68%</i>
	Precision	0.406	0.785	0.394	0.683	0.401	<i>0.683</i>
	Recall	0.411	0.782	0.397	0.685	0.397	<i>0.685</i>
	ROC area	0.554	0.909	0.541	0.836	0.544	<i>0.836</i>

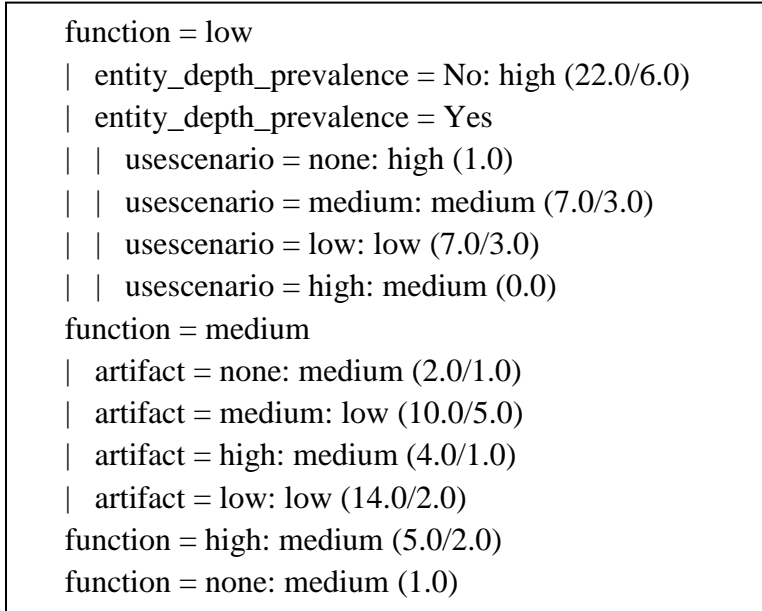


Figure 8.4 Selected decision tree for quality

8.3.5 Examining progress in creativity

The second hypothesis (H2_b) states that novices can be taught how to formulate problems in a way that leads to improved creativity. The models which have been presented so far demonstrate the relationship between problem formulation characteristics and creativity. They suggest how problem formulation characteristics increase or decrease creativity. They do not show if changes in the way several problems were formulated in a chronological order led to improved creativity. To test progress in creativity, the differences in the ideation metrics should be examined across problems assigned to the students in a chronological order. This is done for the participants in group F14G for changes from problems DP_3 to DP_4 and from DP_4 to DP_5. Problems DP_3, DP_4, and DP_5 were assigned to the participants in weeks 2, 6, and 10

of their design course. The differences can be measured in two ways. One is to test the differences for the participants as a whole. The other way is to test the differences for the individuals. This is done with paired *t* test. While the first method assumes that the two samples are independent, the second method assumes that they are collected in pairs. The reason why both methods are of interest here is because two different assumptions can be equally valid. It can be assumed that participant's conceptual design behavior changes in time. It is difficult to control for human factor; people change [90]. Therefore, it is not incorrect to consider that problem formulation and ideation characteristics in the three problems are independent samples. On the other hand, it can be argued that conceptual design behavior is an individual characteristic [20, 34]. Therefore, it is reasonable to consider that the results of the three problems are paired and changes are due to other factors, e.g., effectiveness in learning problem formulation. The results of both analyses are shown in Table 8.10 and Table 8.11. They suggest that quantity and variety increased from the second to the third problem though variety was lower in the second problem compared to the first.

Table 8.10 Changes in ideation metrics for a class as a whole

Ideation metric	DP_4 – DP_3			DP_5 – DP_4		
	Difference	95% CI	P-value	Difference	95% CI	P-value
Quantity	-0.56	-1.74, 0.62	0.34	0.81	-0.36, 1.98	0.17
Variety	-1.92	-3.1, -0.74	0.00	1.87	0.65, 3.12	0.00
Novelty	-0.08	-0.78, 0.62	0.82	-1.27	-2, -0.55	0.00
Quality	-0.86	-1.15, -0.56	0.00	-0.5	-1.05, 0.05	0.07

Table 8.11 Changes in individuals' ideation metrics for a class

Ideation metric	DP_4 – DP_3			DP_5 – DP_4		
	Mean diff.	95% CI	P-value	Mean diff.	95% CI	P-value
Quantity	-0.39	-1.51, 0.74	0.48	0.81	-0.03, 1.64	0.06
Variety	-1.72	-2.79, -0.64	0.00	1.89	0.76, 3	0.00
Novelty	-0.12	-0.89, 0.65	0.74	-1.27	-2.18, -0.37	0.01
Quality	-0.84	-1.16, -0.52	0.00	-0.5	-1.05, 0.05	0.07

There was an overall decrease in novelty and quality of the participants. The drop in novelty can be attributed to the way novelty is computed. If more designers come up with more ideas including ones that would have been novel compared to a larger historical sample, all participants would have a lower novelty score. The drop in the quality score may be attributed to two reasons. One is that the problems which were assigned to the participants as they moved on were more constrained. The other reason is that the participants became more conservative and self-constraining since they were supposed to conduct feasibility study and simulation for the second problem and build a prototype for the third problem.

To summarize, based on the results of the various analyses conducted for Experiment II, the following conclusions can be drawn:

- Quantity may increase if designers do more abstraction, follow a breadth order from adding requirements, and specify key issues without decomposing them (see Table 8.2 and Table 8.3), but it may decrease if designers ignore the relations that functions have to other entities (see Figure 8.1).

- Variety may also increase if designers do more abstraction and specify key issues without decomposing them (see Table 8.2), and decompose use scenarios (see the negative correlation to isolated use scenarios in Appendix D), but it may decrease if designers focus on adding more requirements and use scenarios and identifying conflicts (see Table 8.2, Table 8.3, and Figure 8.2).
- Novelty may increase if designers: a) specify fewer requirements (see Figure 8.3) but more use scenarios and functions (see Appendix E), b) structure more hierarchies especially in use scenarios (negative correlation with isolated use scenarios in Table 8.2) and behaviors (see Figure 8.3), c) recognize issues in relation to other entities (see Figure 8.3), d) follow a depth exploration strategy (see Table 8.2 and Table 8.3).
- Novelty may decrease if designers: a) fail to relate functions to other entities (see Table 8.2), b) identify more conflicts (see Table 8.3).
- Quality may increase if designers specify more behaviors and fewer artifacts, identify more conflicts (see Table 8.3), and follow a breadth exploration strategy (see Figure 8.4). Quality may decrease if designers ignore the relations which requirements have to other entities (see Table 8.2), and the relations which issues have to other entities (see the negative correlation coefficients between disconnected issues and average quality in Appendix E).

In addition, hypotheses H2_a and H2_b were examined. The conclusions summarized above facilitate testing hypotheses H2_b:

- Hypothesis H2_a stated that depth-first exploration of problem formulation entities leads to more creativity. The results showed that depth-first exploration of entities increased all creativity metrics though it had a greater effect on novelty and quantity. Therefore, hypothesis H2_a is proven.
- Hypothesis H2_b stated that creativity can be improved in novice designers by teaching them characteristics of good problem formulation. The results of Experiment I in section 7.3.2 showed positive trends in novices' specification of issues, use of the abstraction strategy, and the entity depth exploration. The results of Experiment II showed a positive trend in quantity and variety and a negative trend in novelty. They also showed how the aforementioned problem formulation characteristics and ideation metrics are correlated. The simultaneous positive trends in the formulation characteristics and creativity metrics (which are statistically significant), their correlation, and the precedence of problem formulation to ideation imply that quantity and variety improved as the novices learned how to formulate problems more effectively. Therefore, hypothesis H2_b is proven for quantity and variety but not for novelty and quality.

CHAPTER 9

EXPERIMENT III: PREDICTING CREATIVITY FROM FORMULATION

The second experiment identified the relationship between problem formulation and creativity in terms of regression models and classification models. These models pave the way for answering the third research question which is if creativity can be predicted from problem formulation. This is the objective of the third experiment. More specifically, the last hypothesis will be tested. The hypothesis states that:

H3) Creativity in design outcome can be predicted with an acceptable degree of confidence from problem formulation behavior.

The mathematical models of the relation between problem formulation characteristics and ideation metrics can be examined for generalizability. The models can predict outcome for new observations. The predictions can be compared to an actual value. In an ideal model the difference is zero. In reality, the differences can be considered random variables for which the null hypothesis $H_0: \mu_D = \mu_1 - \mu_2 = 0$ can be tested. The actual values of creativity, i.e., ideation metrics can be found for different problems. The models built based on one problem can be used to predict the creativity metrics for another problem. The differences between actual and predicted scores can be examined with paired t test. The test also provides determining the level of confidence for the predictions.

9.1 Collected data

The data for this experiment is the same data collected from students of group F14G as Experiment II. However, the first problem is not considered. The reason for excluding the

first problem was to block experimenter's bias. The ideation metrics of the first two problems were scored by the same researchers. To remove bias, the ideation metrics for the third problem (DP_5) were found by an independent panel of judges. The linear regression equations which were described in section 8.3.2 and 8.3.3 are used here to make predictions.

9.2 Analysis method

The methods pertinent to this experiment have been explained in previous experiments. Regression analysis is used to build the models. The differences between the predicted and the actual scores are examined with the paired t test. It should be noted that the paired t test is more powerful than a two-sample t test in design of experiments with fewer observations in the data set. This is because the two-sample t test includes additional variations occurring from the independence of the observations. Observations in a paired t test are dependent. An additional benefit of t tests in general is that they are relatively insensitive to the assumption of normality [106].

Besides test of differences with paired t test, descriptive statistics is used to represent the differences between the actual and the predicted creativity. The distribution of the differences is shown with histograms. Whether or not the successive models which were built during the backward elimination process had any effects on the predictions, boxplots of the differences between actual and predicted scores are shown for successive regression models.

9.3 Results and conclusions

The differences between the actual ideation scores given by the panel of judges and the predicted outcomes from the three models were recorded. The linear regression models are the same that were derived in 8.3.2 and 8.3.3. For example from Table 8.3, the following model can be written for quantity with respect to strategies counts:

quantity (based on DP 4 data)

$$\begin{aligned} &= 2.31 + 0.73 * \textit{abstraction} + 0.18 * \textit{entity depth prevalence} + 3.67 \\ &* \textit{order req use} - 1.65 * \textit{order req fun} - 0.82 \\ &* \textit{conflict identification} \end{aligned}$$

The ideation metrics have normalized scores (on a scale of 1-10). The number of observations which were predicted within 1 or 2 units (10% or 20% margin of error) was reported as the accuracy of the prediction. For one of the participants who used the above strategies 2, 3, 1, 1, 0 times during the formulation of DP_5 the model predicts a score of 6.33. The actual score given by the judges was 2.2 which means a 4.13 difference on a scale of 1-10 (that is more than 20% error). For another participant, the occurrences were 2, 4, 0, 0, 0 respectively which result in a predicted score of 4.49; the score given by judges was 4.6 which is within 10% margin of error of the prediction. Table 9.1 summarizes how accurately the models for DP_4 predicted each of the ideation metrics for DP_5. It can be seen that predictions of variety, average and max quality were highly accurate in models based on state counts and strategies. The strategies counts model is

slightly more accurate in predicting quantity, average novelty, and average quality (within 10% margin of error).

Table 9.1 Accuracy of predicting DP_5 ideation with DP_4 regression models

Ideation metric	State counts		Strategies counts	
	20% error	10% error	20% error	10% error
Quantity	52	32	60	48
Variety	88	64	76	60
Avg. novelty	76	40	64	48
Max novelty	56	40	60	32
Avg. quality	92	72	96	64
Max quality	92	68	92	76

The results of the models after backward elimination are shown in Table 9.2. Compared to prediction accuracies reported in Table 9.1 it can be seen that the predictions in the models after backward elimination are less accurate with the highest drop in the model of variety (from 88% and 64% within 20% and 10% margin of error respectively in the state counts model of DP_4 predicting DP_5, to 40% and 24%). Overall, the prediction accuracy for DP_4 scores based on the DP_5 model is worse. This is partly due to less variation in the ideation metrics of DP_5 among the student participants (F14G group). The results also suggest that an improved model fit does not necessarily guarantee higher predictability for newer observations. Yet, the models after backward elimination showed an overall improvement in prediction accuracy. This is because the definition of accuracy used in Table 9.1 and Table 9.2 is narrow. It disregards

large residuals. Examples of change in prediction accuracy during the backward elimination process are given in Figure 9.1, Figure 9.2, and Figure 9.3. The results are mixed. Quantity predictions improve but the model of quality for DP_5 does not change the prediction accuracy of DP_4. The initial quality model is accurate itself.

Table 9.2 Accuracy of predicting ideation after backward elimination

Ideation metric	Predicting DP 5 from DP 4		Predicting DP 4 from DP 5	
	20% error	10% error	20% error	10% error
Quantity	56%	28%	24%	12%
Variety	40%	24%	32%	16%
Avg. novelty	52%	48%	40%	32%
Max novelty	56%	40%	28%	24%
Avg. quality	76%	56%	36%	20%
Max quality	84%	68%	40%	24%

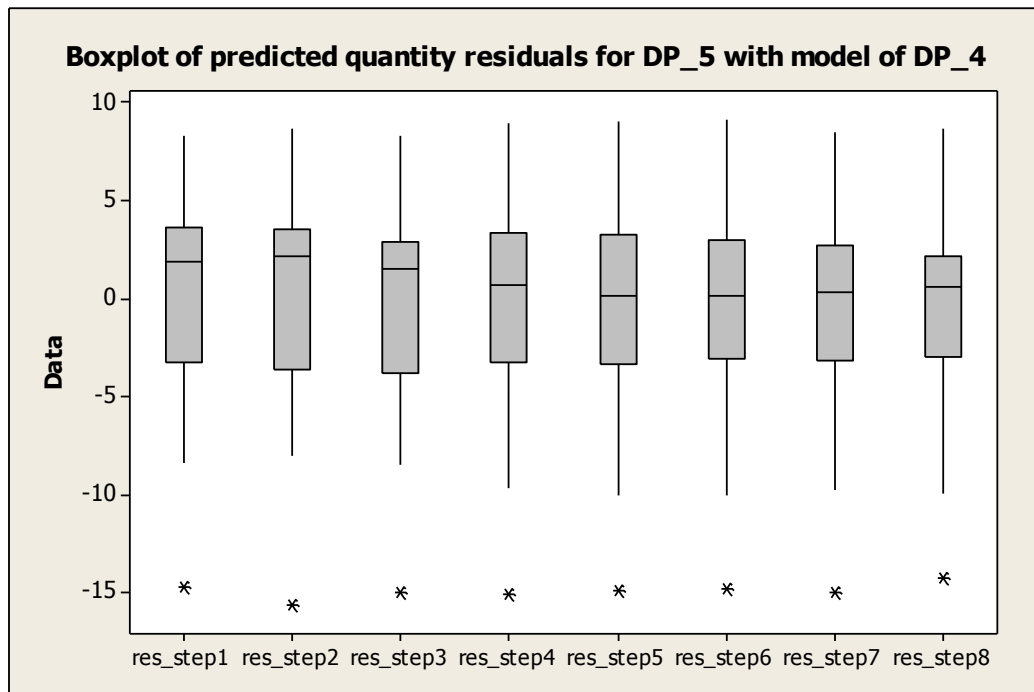


Figure 9.1 Predicted quantity in backward elimination for DP_4 models

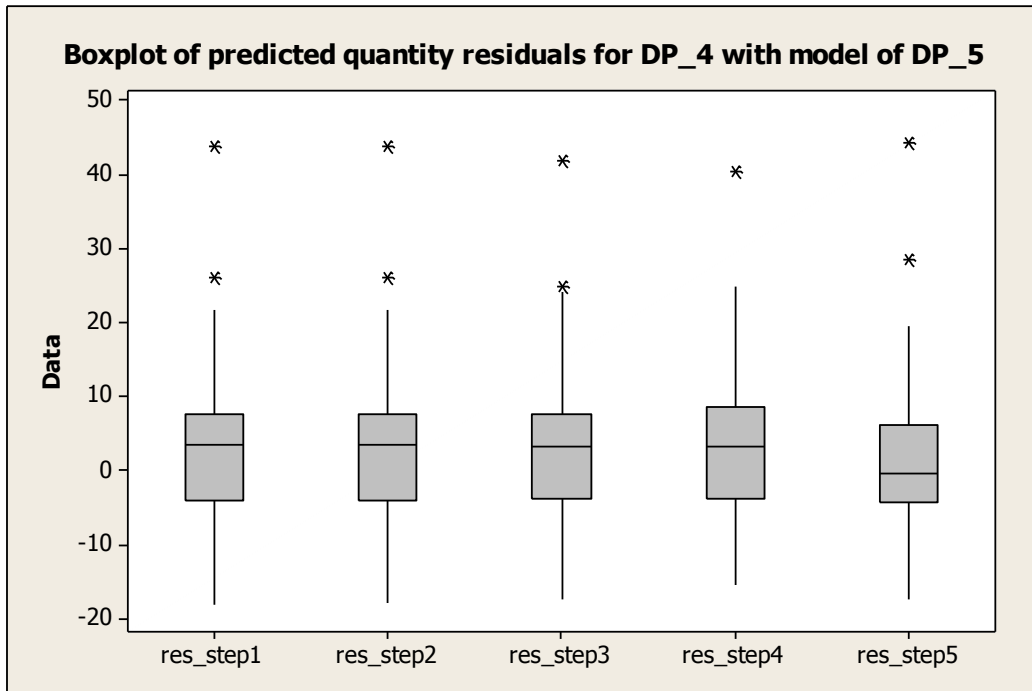


Figure 9.2 Predicted quantity in backward elimination for DP_5 models

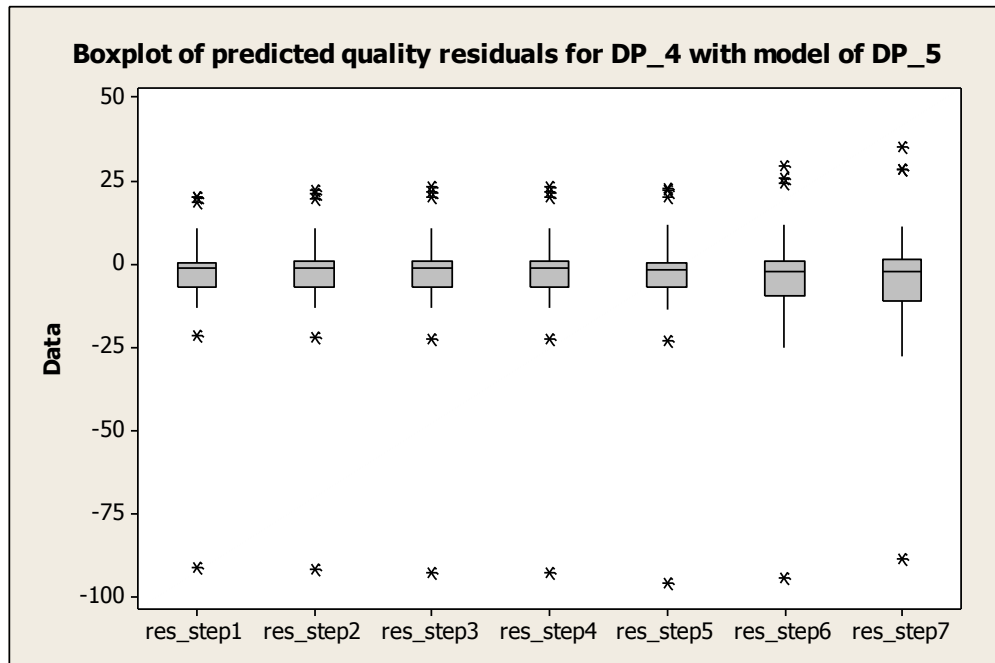


Figure 9.3 Predicted quality in backward elimination for DP_5 models

For Experiment II, different regression models were built with different independent variables. Three models were described: state counts, strategies, and backward elimination on combined problem formulation. The variables in the backward elimination model were normalized (subtracting mean and dividing by standard deviation of each variable) to build a fourth set of models; correlation coefficients in linear regression are invariant to linear transformation. The prediction accuracies of these four models are represented with histograms in Appendix F. Three examples can be seen in Figure 9.4, through Figure 9.6. Models of DP_4 led to more accurate predictions of DP_5 than the other way round. State counts models are also more accurate in prediction, though they have a poor model fit compared to backward elimination of combined normed variables. Predictions with normalized variables are also not too different from non-normed ones.

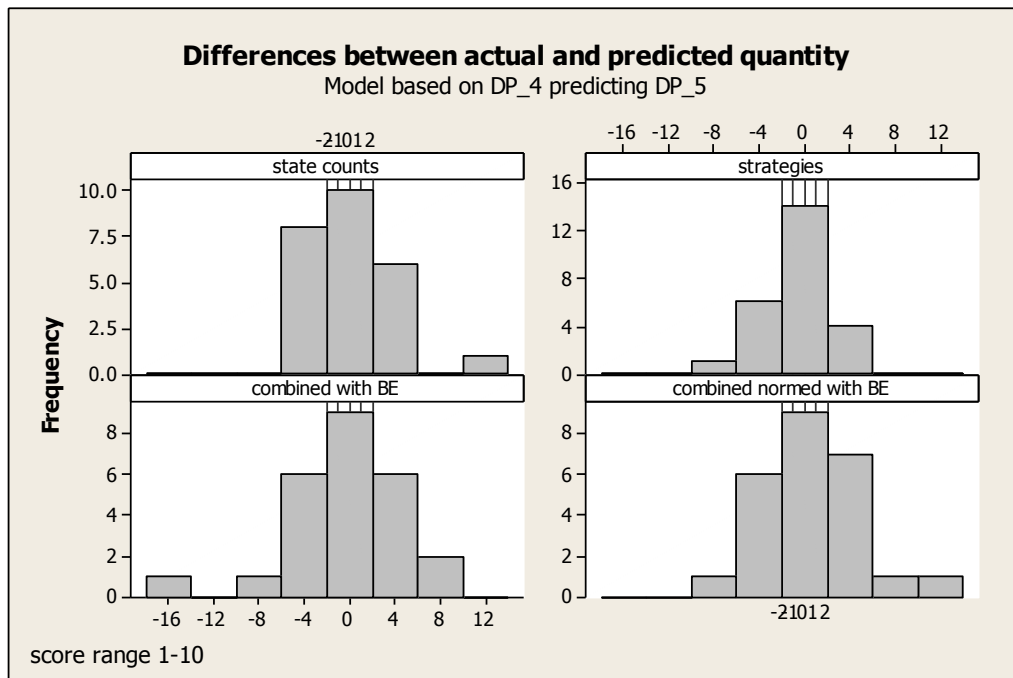


Figure 9.4 Prediction residuals for different DP_4 models of quantity

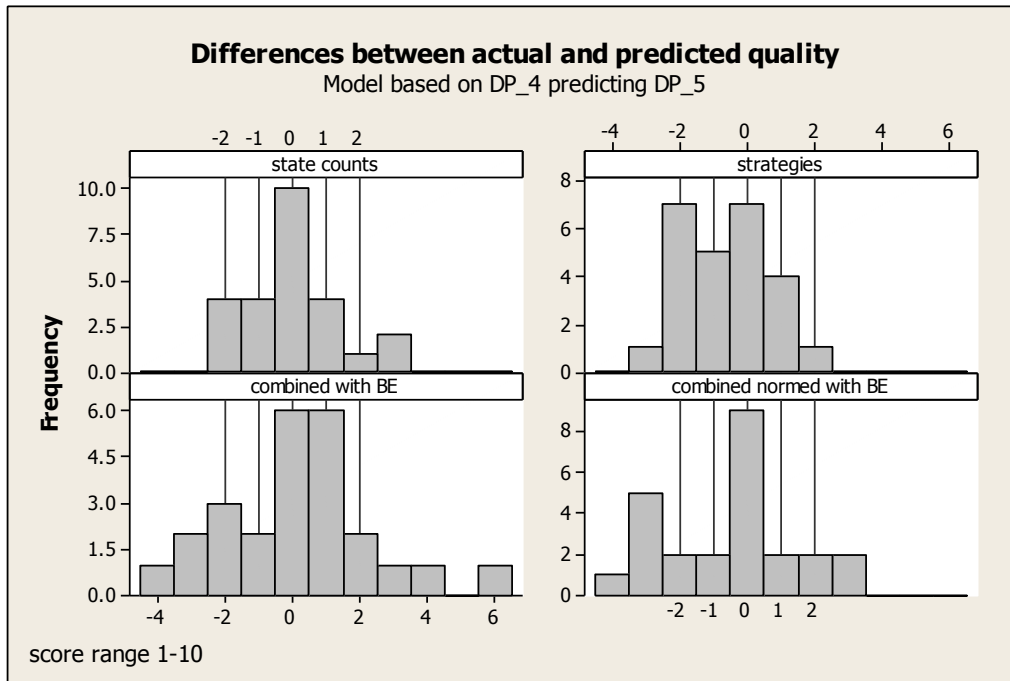


Figure 9.5 Prediction residuals for different DP_4 models of quality

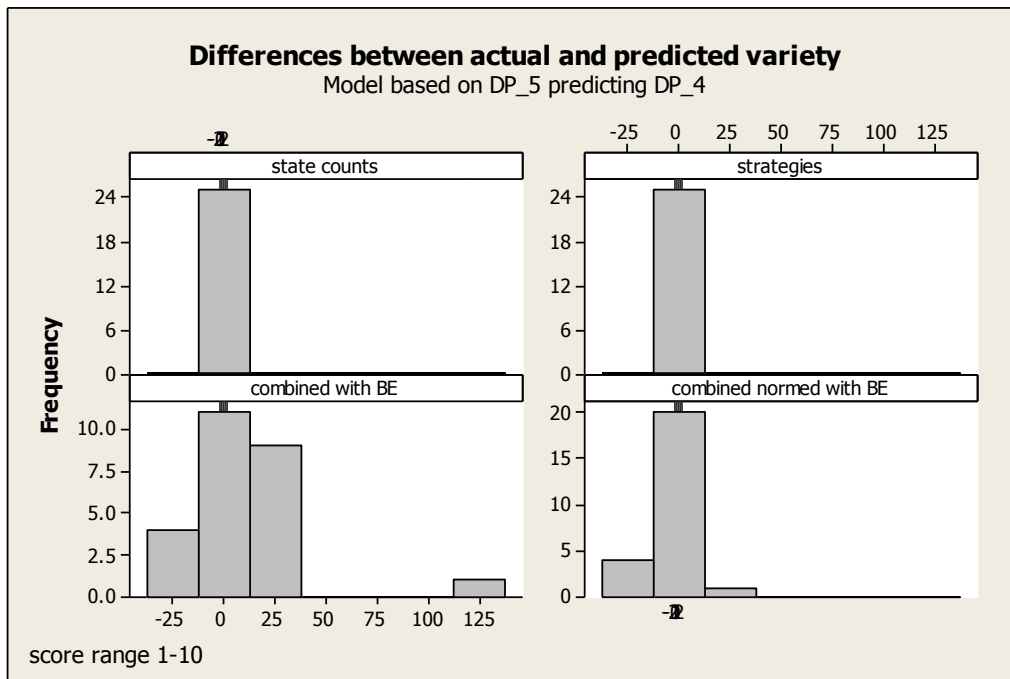


Figure 9.6 Prediction residuals for different DP_5 models of variety

Finally, to examine the differences between actual and predicted creativity (ideation) for statistical significance in relation to hypothesis H3, paired t test was conducted. The results are shown in Table 9.3. For each ideation metric, the mean difference of actual and predicted scores is reported with the corresponding p values. The results suggest that Problem DP_4 cannot predict DP_5 with a 95% confidence (the results are the same even at an 80% confidence level). On the other hand, quantity, variety, and quality of DP_5 are predicted from DP_4 models of combined variables after backward elimination. The histograms of prediction residuals of the four models were misleading for the state count models. The differences in the state counts models might seem smaller (e.g., the mean differences for DP_5 quality model is 4.71 for the state counts model and 9.23 for the combined model). However, higher variability in the residuals leads to rejecting the hypothesis that the difference in the means is zero.

Before finishing this chapter, it should be noted that there is a difference between model transfer and model generalization. What was proposed can be considered a weak model transfer. Model transfer is about examining whether a model of a phenomenon leads to comparable results in a new setting completely independent of when the initial model was built. In the case of Experiment III this would be if the models derived for one problem and group of participants were used to predict data for a completely different problem with different participants. If only one of these two factors (problem and participants) were changed the prediction study would be a matter of generalizing one model to another problem, not transferring it to a new case. However, with human subjects who are learning a task, and with time, the subjects change and they are no

longer the same. In examining hypothesis H2_b it was shown that students' problem formulation characteristics changed over time during a semester. Therefore, it can be said that the problem was changed, the judges who generated the data were different, and the tasks were assigned about 2 months apart. Nonetheless, the prediction results cannot be invalidated and there evidence is provided on statistical significance of prediction accuracy with paired *t* test.

Table 9.3 Differences of actual and predicted ideation; mean row 1; *p* value row 2

	Model	States	Strategies	Combined with BE	Combined normed with BE
DP_4 predicting	Quantity	3.79	4.5	4.6	3.26
		0.00	0.00	0.00	0.00
DP_5	Variety	3.5	5.77	3.9	3.96
		0.00	0.00	0.00	0.00
	Novelty	3.92	5.65	3.5	4.82
		0.00	0.00	0.00	0.00
Quality	4.55	5.15	4.33	5.06	
	0.00	0.00	0.00	0.00	
DP_5 predicting	Quantity	2.88	4.48	0.18	4.06
		0.00	0.00	0.94	0.11
DP_4	Variety	4.37	6.4	-4.81	5.84
		0.00	0.00	0.38	0.03
	Novelty	3.16	3.23	2.67	3.55
		0.00	0.00	0.01	0.00
Quality	4.71	4.83	9.23	4.56	
	0.00	0.00	0.05	0.08	

To summarize, based on the presented results in Experiment III, the following conclusions can be drawn:

- Some creativity metrics may be predicted from problem formulation. This proves hypothesis H3 with some considerations: a) predictions are reliable for models of specific problems, b) backward elimination results in more statistically significant predictions, c) novelty is more difficult to predict due to lower variability when designers become more competent.
- Predictions of variety and quality are more accurate within small margins of error.
- Predictions of novelty and quality are more accurate within small margins of error after backward elimination.
- Some problem formulation characteristics might be invariant to design problems, i.e., they do not require normalization.

CHAPTER 10

POTENTIAL APPLICATIONS

The three previous chapters described three experiments in search for answers to the research questions and to examine stated hypotheses. From the findings of the experiments opportunities arise in using the P-amps framework for potential applications. Three applications are discussed in this chapter. One application is the creation of an applied test of design problem formulation skills. A second application is to use P-maps for an objective assessment of students' problem formulation in design education by defining a set of Problem Formulation metrics. If these two applications seem similar, they are. An analogy can be made to the relation between these two tests and that of the Divergent Thinking test [99] and the ideation metrics [24]. While the Divergent Thinking test and the tentative Problem Formulation test measure a potential skill with a set of questions, the ideation metrics and Problem Formulation metrics assess the outcome of the ideation and problem formulation processes respectively. There is also another application that has a potential to open new avenues in design research. It is to use the framework to examine previous observations and findings from other researchers.

10.1 Applied test of problem formulation skill

Shah [112] had identified a different set of conceptual design skills which a successful designer should possess. A skill is defined as a cognitive ability to perform an engineering design task. A battery of tests have been developed for measuring these skills: divergent thinking [99], and visual thinking [113], qualitative reasoning [114, 115]. A test for problem formulation (PF) has not been developed yet. While the medium

for developing and taking the first three tests was restricted to pen and paper, the PF test can take advantage of the Problem Formulator testbed [96] for data collection and test taking. Another advantage that can be exploited in developing the test is to use the findings from the empirical studies conducted in this research and reported in the previous three chapters to identify problem formulation skills that influence creativity. The process of developing the test, involves identifying sub-skills, defining metrics for measuring each sub-skill, proposing questions and candidate test items, conducting pilot tests and determining which test items lead to a more appropriate distribution of scores for identifying how differently designers possess the skills. The current work is a preliminary task towards the finished test.

10.1.1 Identification of subskills

To identify problem formulation skills two sources can be used: one is the reviewed literature; the other is the findings of the three experiments. The conducted empirical studies explained in previous chapters highlighted the relation between problem formulation characteristics and creativity, more specifically as a list of formulation characteristics influencing ideation metrics in Experiment II. The identified sub-skills and their justification in light of the results of experiments are summarized in Table 10.1. Each sub-skill is discussed below.

A design problem often starts with a problem statement where some customer needs are explicitly stated. The designer must then discover implicit requirements that are necessary to meet. These implicit requirements can be additional requirements at the top level, or derived, as existing requirements are decomposed further. Results of Experiment

II showed that identifying requirements and their relations to other entities led to an increase in quality. This subskill is *requirement elicitation*.

Relationship identification among different aspects of the problem is another subskill that affects creativity in problem formulation. There were several evidences in Experiments I and II about the effect of identifying relations among entities. The results showed that failing to identify the relations to functions and issues adversely affect quantity and novelty respectively. Recognizing the relations that issues had to other entities increased novelty and quality.

Questions about missing information can be defined in P-maps as a subtype of *issue*. Many of the *issues* that the participants raised in the empirical studies were about missing information (e.g., what is the stiffness of the surface where the goofy gopher competition in DP_3 is played). In Experiment II, it was shown that the addition of issues positively influenced quantity and variety. This relates to an *information seeking* subskill.

Table 10.1 Problem formulation skills in relation to creativity (from Exp. II)

Formulation characteristic	Affected creativity metric
Requirement elicitation	Quality
Relationship identification	Quantity, novelty, quality
Information seeking	Quantity, variety
Use description	Novelty
Key objective identification	Quality
Challenging issue	All metrics
Delight addition	Quality
Specification	Quality
Decomposition	Quantity, variety, novelty

One of the causes of bad designs is that designers fail to consider who uses the end product and how. Results of Experiment II showed that specifying more entities about use scenarios increases novelty. The ability to identify use scenarios, or *use description*, is another subskill in problem formulation.

One characteristic of formulating a design problem is to understand where one should pay the main attention to, as resources are limited in a design project. One of the main differences between experts and novices is that experts quickly identify the key objective and the challenging issues, while novices treat everything equally [14]. The related subskills are called *key objective identification* and *challenging issue* respectively. Results of Experiment I also showed a progress in novices' time of identifying issues. Issues were discovered earlier in the final problem (DP_5) compared to the one before it (DP_4). Results of Experiment II showed that identifying the main objectives (under requirements) affect quantity.

One of the aspects that makes good designers stand out is the ability to deliver surprising features in the design that delights customers. The well-known Kano model [116] differentiates between basic features and features of delight in a design where the mere presence of the latter increases customer satisfaction. These feature can be described under requirements in P-maps. The empirical study in Experiment II showed that identification of these requirements increases quality. This subskill is *Delight addition*.

In the same way that problem and solution spaces co-evolve during design and cannot be separated, PF skills involve convergence in addition to divergence. An aspect of

defining the problem is *specification*, setting the boundaries of variables, constraints, etc. Specs are part of requirements in the P-maps ontology which positively relate to quality.

Designers not only expand and bound the design space during problem formulation, but also structure the space. The findings in Experiment II showed that decomposing functions increased novelty and quantity while decomposing entities use scenarios increased variety. The *decomposition* of the problem is also an important subskill in problem formulation.

10.1.2 Associating P-maps measures with sub-skills

Measures can be defined for the identified PF skills. The number of added requirements that are necessary to achieve but implicit, i.e., not directly mentioned in the problem statement, can indicate *requirement elicitation*. The number of identified relations between different fragments of the problem can be a measure of the *relationship identification* skill. The number of times that a designer requests additional information that are important in the design and not apparent in the problem statement, or refers to external sources of information that are known to the designer are indicators of *information seeking*. *Use description* can be measured by the number of times the designer identifies pertinent environmental variables or user affordances. The number of identified key issues and the degree to which the designer allocates resources to them can measure how successful they are at finding the *challenging issue*. The number of auxiliary features of delight that are added can indicate *delight addition*. The portion of parameters that are bounded with absolute or relative ranges and targets constitute a measure of *specification*. The level of decomposing different aspects of the problem, e.g.,

the depth of an objective tree or number of disjunctive functional decompositions, can be indicators of the *decomposition* skill.

Measures within the P-maps ontological framework can be associated with the measures defined above for each problem formulation sub-skill. For example the number of derived requirements can be for the requirement elicitation sub-skill. Table 10.2 proposes a set of P-map measures for the PF subskills. This relation only shows the corresponding measures that one can calculate from a P-map; it does not specify a scoring or grading schema.

Table 10.2 P-maps measures for PF subskills

Formulation characteristic	Problem Map measure
Requirement elicitation	Number of requirements not specified in the problem statement
Relationship identification	Total number of linked entities in all groups in log 6
Information seeking	Number of questions (subtype of issues)
Use description	Total number of use scenarios
Key objective identification	Number objectives (subtype of requirements)
Challenging issue	Total number of issues
Delight addition	Number of delight features added under requirements
Specification	Number of specs (subtype of requirements)
Decomposition	Sum of hierarchy depth and disjunctive branches in the function tree

Scoring the skills can be based on comparing participants' responses to a normative P-map for the given question. The norm can be created from an aggregate of all the P-maps in the same sample or in a historic sample. For example, for scoring the key issue identification subskill one can create an aggregate of possible issues for the given

problem and assign the highest score when the test taker includes all the issues on the list in his P-map, and proportionally lower scores for fewer issues.

10.1.3 Candidate test items

The examples shown in this chapter were for complete design projects. Questions in a test can be in a more controlled setting. Possible questions for different parts of the test can be proposed. Similar to the previous examples, aggregate of responses can be turned into inventories for comparison of the test takers' responses to the norm. One of the important characteristics of a test is how well it reflects on differences among takers. In order to have a test with an appropriate distribution, the questions should be balanced, i.e., most subjects should be able to answer easy questions; some subjects respond better to more difficult questions; a few find the most difficult answers. Finding distributions similar to what was shown in Figure 10.1 helps with that regard.

The main difference between using P-maps for the test and using it for evaluating problem formulation outcome is that in the former, instead of a design task, the assigned questions are limited to measuring one or a few of the characteristics. For example, a question can ask the test taker to pick the order of issues which are more challenging in a specific design situation from a provided list of issues for that problem. Another example is testing decomposition in two ways. One is to given a high level function and ask the test taker to provide as many functions as possible in lower levels. A different way of posing the question is to provide an incomplete function structure and ask the test taker to fill in empty nodes.

The test has not been fully developed yet but its structure and some candidate test items are discussed in [117]. Unlike the previous design skill tests [99, 113, 118], this test is planned to be taken on a web-based testbed, not pen and paper, with the intention that more subjects take the test and be graded quickly. One remaining challenge in using the tool for this purpose is that some automatic text processing is required for assessing the free form text responses collected in P-maps.

10.2 Objective evaluation of students' problem formulation

Since the P-maps framework facilitated data collection about problem formulation in a structured way, it was feasible to find a rubric from the diverse set of variables which P-maps provide for evaluating problem formulation skills. To compute some of the formulation variables, inventories should be created based on all the responses from all the students, similarly to how it is done for calculating ideation metrics. The process was described in section 6.4. Table 10.3 and Table 10.4 show examples of implicit and fictitious requirements, and key and irrelevant issues for the goofy gopher problem (DP_3) problem derived from an aggregate of all the P-maps.

Table 10.3 Examples of implicit and fictitious requirements inventory

Implicit	Fictitious
should collect balls with higher points	should store few balls in device
should protect collected balls from the opponent	should carry ball to silo
should endure the whole tournament	should minimize weight
should sustain impacts from opponent's device	should move back and forth
should be easy to control by one operator	should not have excess cables

The type of responses from entries into the testbed within the defined categories of the P-maps ontology was determined by two judges through a process of arbitration. One major factor in deciding if a response is appropriate is to see if it unnecessarily bounds the design space at such an early stage. For example the implicit requirement ‘should store few balls in device’ implies a certain design where the device moves on the field, while a viable design option is to deliver the balls from the point they are picked to the silo without carrying them, e.g., by throwing.

Table 10.4 Examples of key and irrelevant issues inventory

key issues	irrelevant issues
Control of the device with one operator	Mechanism degrees of freedom
Material constraints limit variety of solutions	Interfering with opponent's device without damaging it
Managing power consumption	

To objectively assess students’ problem formulation, a grading schema was set up for the identified P-maps measures. Table 10.5 shows this grading schema. The measures are normalized with respect to the sample to create a scale of 1-10 similar to the scales in the applied design skill tests [99, 113, 118] and ideation effectiveness metrics [24]. Some measures can be found by deducting points when the students choose inappropriate responses. This is similar to how afixability is computed in the Divergent Thinking test [99]. For most measures the response should be appropriate which is determined with respect to the inventories created as explained above. Few scores can be directly measured from raw counts (of problem formulation characteristics). The distribution of the students’ scores is shown in Figure 10.1.

Table 10.5 The scoring scheme for evaluating students' PF in a design task

Subskill	Measure (normed by dividing by max in sample)	Response inventory
Requirement elicitation	Total derived requirements	Yes
Relationship identification	$\log_6 Total\ links$	No
Information seeking	Total questions (sub-type of issues)	Yes
Use description	Total use scenarios	No
Key objective identification	10 – total incorrect objectives with high importance	Yes
Challenging issue identification	10 – total incorrect issues with high importance	Yes
Delight addition	Total derived delight requirements	Yes
Specification	Total number of specs (sub-type of requirement)	No
Decomposition	Width + depth of the function hierarchy	No

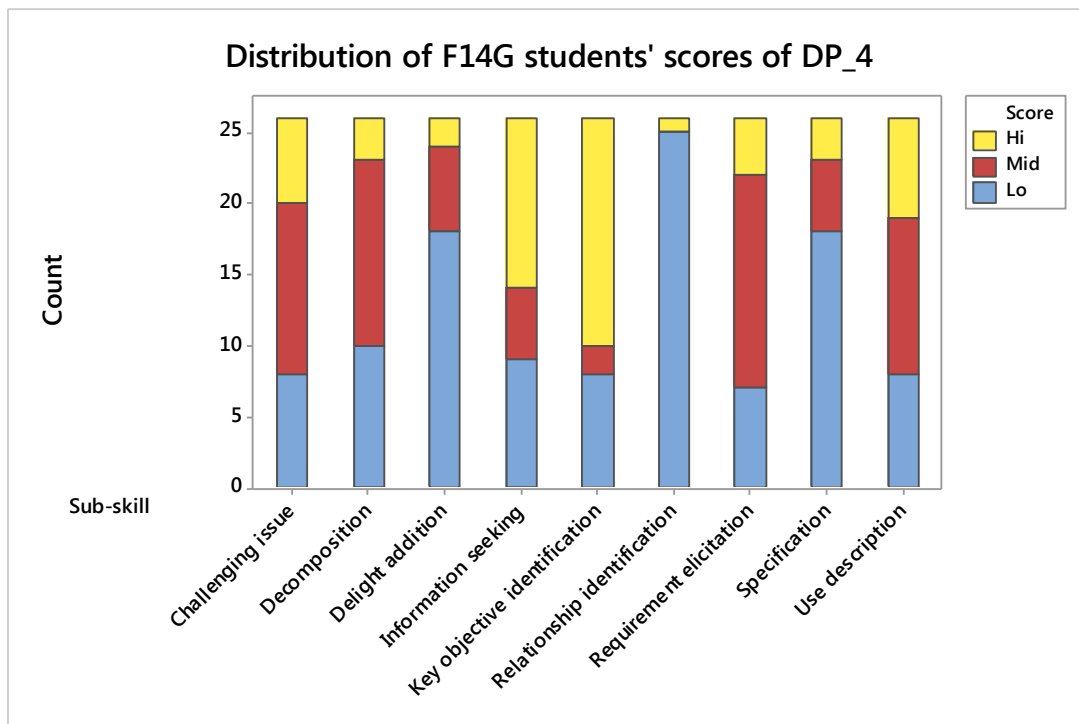


Figure 10.1 Distribution of students' grades of PF skills for a design task

The scores of sub-skills can be measured during an interval to track students' progress. This was done for the participants in group F14G. Scores of students' nine problem formulation sub-skills were compared for two design problems (DP_4 and DP_5). Figure 10.2 shows the changes in problem formulation characteristics from DP_4 to DP_5.. It can be seen that for the majority of the sub-skills, the students not only improved but also converged. Major improvements occurred in finding implicit requirements, identifying the challenging issues, and creating a more comprehensive spec sheet (the specification skill). The decomposition skill also saw improvement; this may probably be attributed to learning how to better use the Formulator testbed. To find out if the aforementioned changes were statistically significant or not, a paired *t* test was conducted to evaluate the differences in the means for each sub-skill. Table 10.6 summarizes the results. It can be seen that use description significantly went down which can be explained by the more constrained nature of the problem. While different user groups and environmental conditions affect the shot-buddy design (DP_4), there are relatively fewer use scenarios for the autonomous surveillance design problem (DP_5).

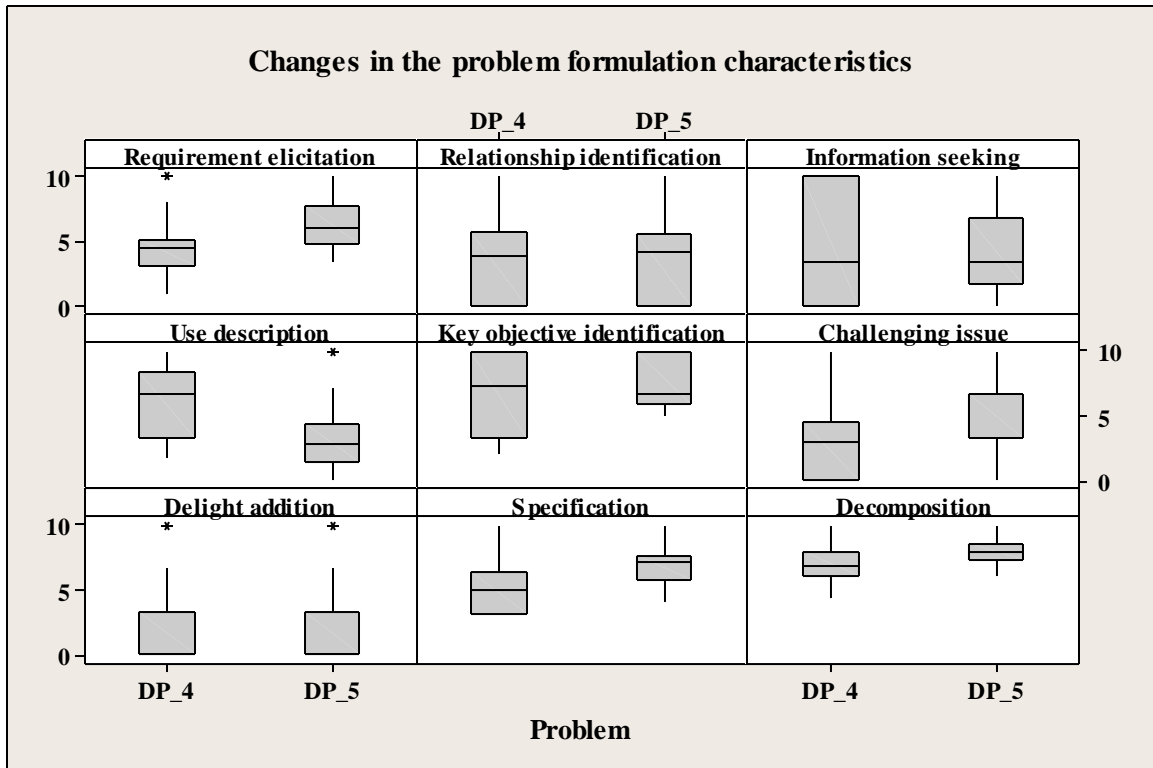


Figure 10.2 Changes in students' problem formulation characteristics

Table 10.6. Test of changes in individuals' problem formulation sub-skills

	DP_3-DP_5	p-value
Requirement elicitation	1.96	0.00
Relationship identification	0.34	0.56
Information seeking	-0.97	0.32
Use description	-2.86	0.00
Key objective identification	0.86	0.27
Challenging issue	1.94	0.02
Delight addition	0.14	0.85
Specification	1.53	0.00
Decomposition	0.92	0.01
Total score	0.68	0.02

10.3 A vehicle for reproducing previous studies

One of the main motivations in breaking away from protocol analysis in this research and embracing the application of a computer testbed based on an ontological framework with a limited set of defined entities has been to enable large scale data collection and analysis. Research in design thinking often suffers from studies with small sample sizes [2]. A consequence of this difficulty in conducting empirical research in design is that unlike some fields in science such as material science or even marketing in humanities, studies with the objective of reproducing previous findings are almost non-existent. An indirect advantage of following the proposed method in this research (data collection on computer testbed) is that efforts in replicating previous studies or comparing the variety of the results which were found in this research to the findings in the literature becomes more convenient than if protocol analysis was used.

This section provides a few examples from comparing observations from the literature with some of the discoveries in this research. The first example involves the role of the direction of search and exploration in the design space. While Ball *et al.* [40] stated that experts use breadth-first search when novices use depth-first search, Ho [39] stated that expert designers use depth-first exploration more successfully. On the other hand, Cai *et al.* [41] found no difference between subjects who follow either depth or breadth exploration of the design space with respect to creativity. In this study, for student subjects, a positive correlation was found between depth-first exploration and novelty.

The second example is about how designers decompose a problem. There have been quite contrasting observations in the literature on the effect of problem decomposition. Liikkanen and Perttula [38] found that decomposition does not affect creativity while the

opposite was found among expert designers [39]. In this study, I found that following specific orders in decomposing different aspects of the problem (adopting the *order req_use* and *order req_fun* strategies) increased the quantity of ideas.

Another example revolves around a well-known strategy in the literature of design; how abstraction influences creativity. Ward et al. [36] described the role of abstraction in improving novelty in ideation. Ball *et al.* [37] also found that experts used abstraction more frequently than novices did. The findings in this dissertation provide a detailed account of how statistically significant the influence of abstraction is on each metric, and if the results stand if a factor such as the design problem is varied. For the two problems DP_4 and DP_5, abstraction was found to positively influence the quantity and the variety of ideas for one problem. There was a positive influence on max novelty and max quality though the correlation was not statistically significant (p 0.52 and coefficient of 0.26 for max novelty; p 0.33 and coefficient of 0.14 for max quality).

Some of the studies that were reviewed in the literature suggest promising alignment between qualitative results with results obtained from quantitative analyses that utilize computational frameworks [41, 47]. I shall emphasize that using computational methods with data collected on a large scale as it was carried out in this research, and coupled with text analysis methods and formal ontologies as will be suggested for future work might help design researchers to reach new findings while avoiding tedious and resource-consuming qualitative research methods.

CHAPTER 11

CONCLUDING REMARKS

11.1 Research questions revisited

The main objective of this research has been the understanding of problem formulation in engineering design and how it may affect creative outcome. There is enough evidence from past studies that experienced and/or creative designers, approach design problems differently from novice and less creative designers. However, the findings have often been at a high level and sometimes contradicting each other. The main hindrance in studying how designers think (or how they formulate problems which is the scope of this research) has been the tediousness of the main method of studying design cognition, protocol analysis. This research has proposed a new method for investigating problem formulation in design; modeling data in a computational and ontological framework which can be collected and analyzed on a large scale in a computer testbed. A variety of quantitative assessment models and qualitative observations of designers were found throughout this work and in adopting the proposed method. Let us revisit the research questions and the stated hypotheses of this thesis:

1. What model can be used to capture a designer's understanding of a design problem, and show individual differences in problem formulation?
2. How do more creative and/or experienced designers formulate design problems differently from less creative and/or novice designers? How can the differences be captured within the framework?

3. Can creative outcome be predicted from the way designers formulate problems?

H1_a) Novice designers follow a systematic order in expressing problem formulation while experts have a more opportunistic behavior.

H1_b) Experts find key issues early on during problem formulation while novices find more issues and later in the formulation process.

H2_a) Depth-first exploration of problem formulation entities leads to more creativity.

H2_b) Creativity can be improved in novice designers by teaching them characteristics of good problem formulation.

H3) Creativity in design outcome can be predicted with an acceptable degree of confidence from problem formulation behavior.

Chapters 3 and 4 covered the answer to the first research question. The Problem Map framework was presented and compared to different modeling frameworks to reaffirm the motivations behind proposing P-maps and the lack of an appropriate ontological framework in past work for studying problem formulation. To evaluate the appropriateness of the P-maps ontology for expressing problem formulation data, one of the common methods was used which is finding inter-rater agreement in assigning fragments to entities in the ontology. The worst agreement was a 0.28 Cohen's Kappa between two previous users of the testbed associated with the ontology. The best agreement was found between two researchers intimately involved developing the ontology at 0.75 Cohen's Kappa which is considered near perfect. In describing how strategies could be formalized and traced in the P-maps framework in 6.3.3, predicate

logic formalism (ASP/Prolog) was also described as a textual representation of problem formulation data.

The answer to the second research question was provided with results of the first designed experiment in chapter 7. Even though the expert designers were a small sample, some differences among them could be observed in addition to differences to the student subjects in this study. One example of a within subject (expert) difference was that most designers added requirements early in their problem formulation while one expert continued adding requirements throughout his work. An example of a difference found between experts and students was that experts had a higher rate of adopting the abstraction strategy than students, while students followed a forward order (defining and relating requirements, functions, artifacts, and behaviors in this specific order) more than the experts did. In addition, hypothesis H1_a was proven but hypothesis H1_b was rejected.

Some of the findings in this research may have been reported to a degree in the past as explained in section 2.1. However, the main contribution of this work though comes from the detailed empirical findings based on correlation analysis, linear regression modeling, and a host of statistical data mining methods facilitated by the fine-grained ontological framework, results of which were explained in Experiment II throughout chapter 8. In addition, the relationship between problem formulation and creativity was studied. Characteristics of problem formulation were related to ideation metrics. The key findings are:

- Quantity may increase if designers do more abstraction and specify key issues without decomposing them, but it may decrease if designers ignore the relations that functions have to other entities.
- Variety may also increase if designers do more abstraction and specify key issues without decomposing them, and decompose use scenarios, but it may decrease if designers focus on adding more requirements and use scenarios and identifying conflicts.
- Novelty may increase if designers: a) specify fewer requirements but more use scenarios and functions, b) structure more hierarchies especially in use scenarios and behaviors, c) recognize issues in relation to other entities, d) follow a depth exploration strategy.
- Novelty may decrease if designers: a) fail to relate functions to other entities, b) identify more conflicts.
- Quality may increase if designers specify more behaviors and fewer artifacts, identify more conflicts, and follow a breadth exploration strategy. Quality may decrease if designers ignore the relations which requirements have to other entities, and the relations which issues have to other entities.

From the results of Experiment II hypotheses H2_a and H2_b were also proven. The answer to the third research question came from using the regression models built for two problems to predict the outcomes for one another. This was covered in chapter 9 as the third designer experiment. Predicted results were compared to scores assigned by an independent panel of judges. The R-squared and R-squared adjusted statistics, as well as the difference between the scores predicted by the models and scores assigned by the

judges were used as indicators of model fit (predictability) and accuracy of the regression models respectively. Models of novelty and quality had statistically more reliable models. Models of variety, novelty, and quality had more accurate predictions. An iterative backward elimination method was used to remove the regressors which were statistically less significant, in order to produce a more reliable model with respect to the R-squared adjusted statistic. Predictability of the models improved significantly (the least change in R-squared adjusted was for max quality from 95% to 97%; the most change was for variety from -201% to 31%). However, accuracy of the predicted outcomes dropped especially for variety.

It should be noted that in retrospect, a few other questions were partly dismissed either because they did not fit the scope of this research, or they were sidestepped in search for answers to more fundamental questions. One of the initial questions was: “Is it possible to build an interactive computer tool that aids problem formulation leading to creativity?”. Obviously, the answer has involved the development of an interactive tool which has been used in this research as a testbed for data collection. Another change from an initial plan of research related to the evaluation of the implemented modeling framework. Instead of evaluating the framework with respect to the initially proposed criteria (domain-independence, richness, compactness, unambiguity, and flexibility), a common approach to the evaluation of ontological frameworks was used: inter-rater agreement. The main reason was that determining measures for the initial criteria set was subjective and uncommon in the literature, but measures for inter-rater agreement are well-established.

11.2 Limitations

11.2.1 Limitations of the exploratory studies

While the P-maps models allows one to represent a large part of the problem formulation process the designers went through, there were some things that could not be coded using the model in the exploratory protocol studies. The reasons lie within the shortcomings of the protocol analysis method. One is that the process relies on the judges or raters' interpretation of verbalized thought. The other is that verbalized thoughts are incomplete [119], i.e., the designer does not express all the process that goes through mind verbally. Examples of such limitations are described in this section. Some of these limitations led to changes in the ontology as described in chapter 3. Implementing a computer testbed instead of a think-aloud method of data collection could overcome other limitations.

One of these limitations was that the model was designed to be domain independent. While this was a major strength of the model, this also meant that without domain knowledge, the different combinations of possible designs that may have been generated from the P-map might have contained artifacts, or other entities that could not combine well or at all in reality. In order to allow for this information to be entered, the problem map model would need to allow the designer to specify when two entities could not be combined.

There was no way to specify whether the children of a parent were both required or if they were disjunctive when interpreting the transcriptions. For example, a device may have either required a regular valve or a one-way valve, or both may have be required in

different parts of the device. These valves would be coded in the following way regardless of whether they were conjunctive and disjunctive:

```
physicalEmbodiment(em_one_way_valve).
```

```
physicalEmbodiment(em_valve).
```

```
parentOf(sl_device, em_valve).
```

```
parentOf(sl_device, em_one_way_valve).
```

This was due to the nature of how these physical embodiments were often introduced in the protocols and the fact that proto-solutions often overlap, sharing many entities. This information could be encoded using the P-map modeling framework, but encoding hierarchical information from protocol studies was prohibitive.

Additionally, in some instances, designers connect components to the high-level solution principle of the device. When a more specific device was mentioned, it might have been the case that the child did contain the components connected to the high level device, or it might have been the case that those high level components were actually connected to a disjunctive solution. This was another piece of information that could not be coded.

Functions specified by the designer might have been used in a sequence multiple times with different parameter values. While the P-map model coded sequential information, there was no way to specify which parameter value went with which instance of the function. For example, one designer's protocol mentioned the ascend function three times during the process of collecting the sample. The first time, the designer wanted the device to ascend ten meters, puncture a balloon, ascend another predetermined amount, collect the sample, and then drop the weights and ascend the remaining distance to the surface.

Another piece of information that was hard to encode was whether a parent solution principle of a physical embodiment was an abstract solution principle guiding the selection of entities, or a parent, which contained the child physical embodiment. For example, one designer specified that the design should incorporate disposable liners for the water-sampling container to avoid contamination between samples. This liner therefore was specified as both a child of the solution principle `sl_disposable` and as a child of the `sl_water_sampler` though these relationships were different. In another example, one designer first specified that he wanted a water container, and that this device should have a balloon. Later he elaborated and said that he wanted a pressure containment vessel as the water container. Both pressure containment vessel and balloon would have been coded as children of the higher level water container.

Another observation was that the coding scheme linked parameters, such as spatial location to the entity the location information belongs to, but not necessarily the entity that it affected. For example, if a solutionPrinciple `sl_device` had an embodiment `em_hatch`, the parameter (`pr_hatch_location`) would be linked to the device, without any sort of link to the `em_hatch`. While this type of information was not necessary for the analyses presented earlier in this paper, it would become more relevant when assessing a formulated problem with measures such as quality, quantity, fluency, and originality of the resulting design outcome.

Finally, the designers were often found specifying information about what did not need to be considered in the design space. For example, one designer concluded that, since the device was intended for freshwater use only, salt erosion, oxidation or any contamination of the materials could be safely ignored. There was no clear way to code

this information. On the one hand, the model allowed for a statement such as “the device should be made out of materials that do not become contaminated and that should be resistant to salt erosion or oxidation.”. On the other hand, it was possible to use negation in the predicate logic formalism of ASP, though unlike a well-defined problem, the ill-defined nature of a design problem with a design space that cannot be finitely bounded does not make the defined problem space with counterfactuals trivial.

11.2.2 Limitations of the experimental studies

Three major challenges were faced in the way that data was collected. The first challenge related to the difficulties that were experienced in using the data collection tool, the web-based Problem Formulator [96]. Similar to any software tool there is a learning curve. Prior to working on the problems which were used throughout this study, the student participants in all groups (F12U, F13G, and F14G) learned about the tool and its underlying ontology in an hour long workshop, in addition to working on a different practice design problem (students in F14G had an additional workshop presenting the depth-first and breadth-first approaches). Yet, some students still misused the tool in entering fragments under the wrong categories. Another common mistake was to mistake conjunctive relations with disjunctive relations (which mean alternatives) under a parent node. A part of future work will be to embed a pre-verification system in the tool where users will be prompted to correct their entries, or a more appropriate category is suggested by the tool.

Another challenge in this study was the limitations of selecting appropriate design problems. Even though the ideation metrics have a normalized scoring schema with

respect to either a historical pool from previous designs for the same problem, or the sample of designers' concepts at hand, it is difficult to find two problems which lead to ideation outcomes of the same distribution of scores. Some problems, by the inherent constraints that they have, lead to less ideas with less variety in the proposed solutions, which in turn lowers the chance of having high scores of novelty. Figure 11.1 shows the changes in the variety scores of the students for the DP_4 and DP_5 problems. Even though the median remains fairly the same in both problems, the distribution is much narrower in the variety scores of DP_5 compared to DP_4. Figure 11.2 shows how average novelty goes down from DP_4 to DP_5, mainly because the second problem was more constrained since the students were asked to build a working prototype to compete with other students. It is plausible to assume that the students became more conservative in proposing their designs merely due to the fact that they were subconsciously searching for a design that worked.

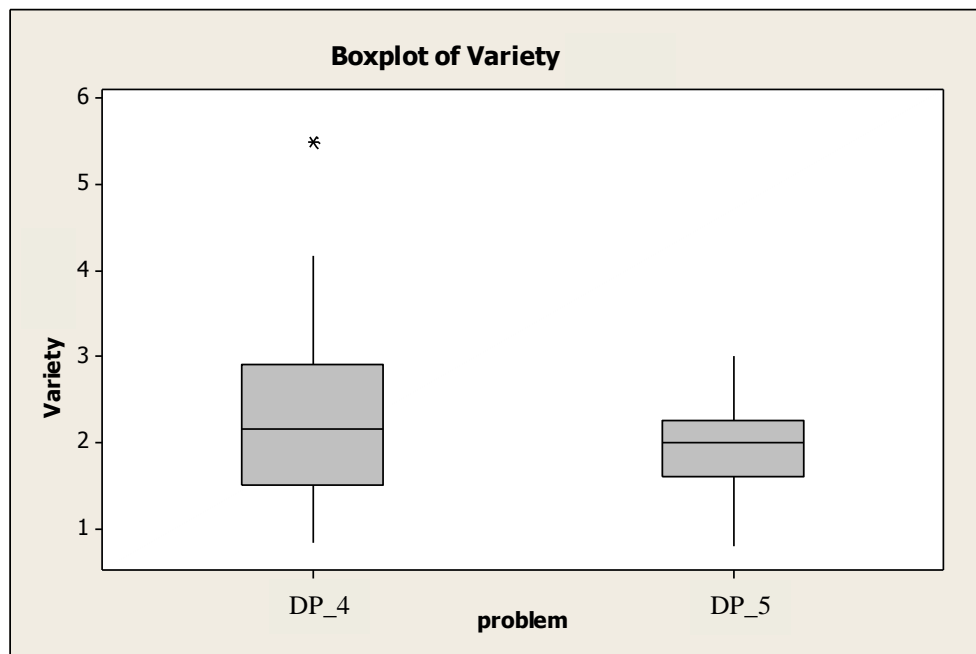


Figure 11.1 Decreasing variability in variety (DP_4 to DP_5)

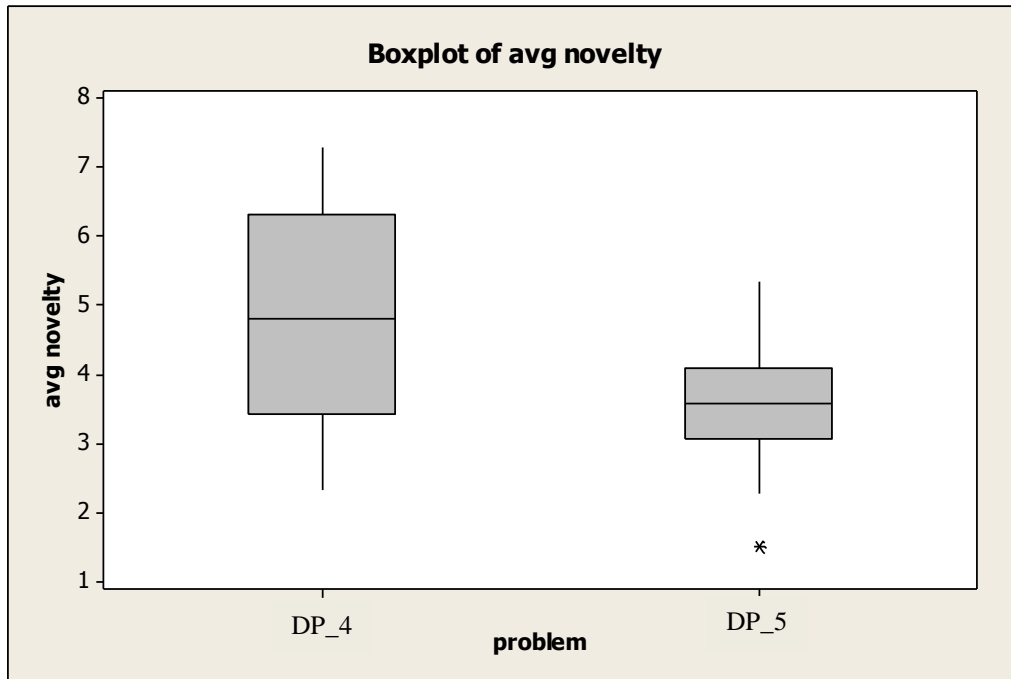


Figure 11.2 Decreasing mean and variability of average novelty (DP_4 to DP_5)

The challenge in problem selection is coupled with the nature of a progressing class of students (who have been the majority of the participants in this research) throughout a semester in further lowering the variation in outcomes. As the class progressed, through multiple assignments and design projects, the students design skills improved, resulting in a convergence in some of the ideation metrics from DP_4 to DP_5. An alternative interpretation of Figure 11.2 is that it was less likely to come up with a novel idea when students' level of competence had become close.

11.3 Future work

The creation of a computational framework based on an ontology and an associated computer testbed for a large scale data collection and analysis is a promising method in research in engineering design and designer thinking. There are four major directions to follow in the future. One is to test hypotheses based on the current observations. For

example, results from the regression models suggested that structuring behaviors improves quality but lowers novelty. To test this hypothesis, one can set up an experiment with a test group that receives recommendations about structuring their behaviors, while a control group is discouraged from doing so (or in alternative experiment, the control group does not receive any recommendations). Similarly to testing hypotheses based on observations from this research, another possibility is to use the framework for testing hypotheses based on past work by others. The fine-grained framework can be used to validate or refute previous findings. Examples were given in section 10.2.

The second direction is to identify more problem formulation strategies either by formalizing them based on introspection or the literature, or by using machine learning methods to propose new strategies. Two possible approaches are using templates and Inductive Logic Programming. With templates, meta-level rules can be defined instead of the specific strategies which were defined in 6.3.4. An example of template can be formulated in the following meta-level rule: given an entity of type A at time 1, find if there are more entities linked to it of type B or type C. Another template can have this meta-level rule: given an entity of type A at time 1, find if entities of type B were linked to it before entities of type C. Inductive Logic Programming [120] combines logical knowledge representation with machine learning in a relational learner, i.e., it takes advantage of a predefined knowledge of relations among attributes or features (a belief network) to generate human-interpretable explanations.

The third possibility for future work is to turn the current computer testbed into a coaching or tutoring system. Based on the measures associated with the problem

formulation skills as explained in Chapter 10, one can diagnose participants' weak problem formulation sub-skills, and provide prescribed recommendations to improve individuals. The development of the problem formulation skill test is also a step towards that goal.

Finally, an important task for the future is to overcome the main challenge in scaling up empirical studies using the associated computer testbed, which is the automation of the understanding, and categorization of the text inputs. One shortcoming of the current testbed is that users can enter data fragments in the wrong categories. Some of the measures described in this research could be found without looking into the data fragments, e.g., the total number of functions. Some measures require understanding the meaning of the fragment, e.g., whether a requirement is implicit or fictitious or whether an issue is a question for seeking information or about a conflict between different requirements. Understanding text fragments is a first step for automatic evaluation of the input. Providing the users with a score also requires creating normative P-maps from an aggregate of a sample to be compared to. Automating this step in the process is even more challenging since one should determine which responses are close in meaning and should fall under one cluster.

11.4 Original contributions

Problem formulation is an important yet understudied subject in designer thinking. Existing frameworks and methods of empirical investigation lack a level of detail appropriate for studying how problem formulation influences creativity. This motivated the creation of a new ontological framework which facilitated answering research

questions about the characteristics of problem formulation in relation to creativity. The Problem Maps framework is one of the original contributions of this research. An earlier version of the P-maps model was also reported as one of the original contributions in the thesis proposal.

One of the main contributions of this research is the creation of a theoretical framework for representing design strategies in a formalized way. There are two benefits in the proposed framework. One is that a fairly qualitative designer behavior is turned into a quantitative variable (counts of occurrences of strategies). Second, strategies are defined as a set of actions that meet certain conditions regardless of any other actions that is happening in an interval as long as they do not violate the conditions of the strategy. The set of strategies defined in this work was small but there is a potential in identifying more strategies as it was explained in the previous section on future work.

A computerized testbed was created to speed up data collection, data analysis, and the rate of discovery of empirical findings. The Problem Formulator testbed was another original contribution of this research. The testbed was used to collect data to conduct experiments to answer the research questions and proposed hypotheses.

Three experiments were designed to understand the differences within and between novices and experts, model the relation between problem formulation characteristics and creativity, and examine if creativity can be predicted from problem formulation. Results of the protocol study with the eight experts were reported in the thesis proposal as an original contribution. Comparisons to novices are additional contributions.

The models of ideation metrics with respect to problem formulation are also original contributions of the research. They led to a list of problem formulation characteristics

which influenced creativity. Based on these relations, recommendations can be made for improving novices' problem formulation skills. Another contribution of this research was to enable predicting a designer's creative outcome based on his problem formulation. Finally, new hypotheses were suggested based on the findings from the empirical studies.

11.5 Publications

Journal papers:

1. Dinar M., Danielescu A., Maclellan C., Shah J. J., and Langley P. "Problem Map: An ontological framework for a computational study of problem formulation in engineering design", *Journal of Computing and Information Science in Engineering*, 15(3), 031007
2. Dinar M. , Shah J. J, Cagan J., Leifer L., Linsey J., Smith S., Vargas-Hernandez N., 2015, "Empirical Studies of Design Thinking: Past, Present, Future", *Journal of Mechanical Design*, 137 (2), 021101
3. Maclellan C., Langley P., Shah J. J., and Dinar M., 2013, "A Computational Aid for Problem Formulation in Early Conceptual Design", *Journal of Computing and Information Science in Engineering*, 13 (3), 031005

Conference papers:

1. Dinar M., Park Y., Shah J. J, Langley P., 2015, Patterns of Creative Design: Predicting Ideation from Problem Formulation, *ASME DETC*, Boston, MA, USA

2. Dinar M., Park Y., Shah J. J, 2015, Evaluating the Effectiveness of Problem Formulation and Ideation Skills Learned Throughout an Engineering Design Course, *ASME DETC*, Boston, MA, USA
3. Dinar M., Park Y., Shah J. J, 2015, Challenges in developing an ontology for problem formulation, *International Conference on Engineering Design (ICEDP_45)*, Milan, Italy
4. Dinar M., Shah J. J, 2015, Towards a Comprehensive Test of Problem Formulation Skill in Design, *The 3rd International Conference on Design Creativity*, Bangalore, India
5. Dinar M., Shah J. J, 2014, Enhancing Design Problem Formulation Skills for engineering design students, *Proceedings of ASME DETC*, Buffalo, NY, USA.
6. Cagan J., Dinar M., Shah J. J, Leifer L., Linsey J., Smith S., Vargas-Hernandez N., 2013, “Empirical Studies of Design Thinking: Past, Present, Future”, *Proceedings of ASME DETC*, Portland, OR, USA.
7. Dinar M., and Shah J. J., 2012, “A Model of Problem Formulation Strategies in Engineering Design,” *Proceedings of First Annual Conference on Advances in Cognitive Systems*, P. Langley, ed., Palo Alto, CA, USA.
8. Danielescu A., Dinar M., Maclellan C., Shah J. J., and Langley P., 2012, “The Structure of Creative Design: What Problem Maps Can Tell Us about Problem Formulation and Creative Designers,” *Proceedings of ASME DETC*, Chicago, IL, USA.
9. Dinar M., Maclellan C., Danielescu A., Shah J. J., and Langley P., 2012, “Beyond Function-Behavior-Structure,” *Design Computing and Cognition*

DCC'12, J.S. Gero, ed., Springer, Texas A&M University, College Station, TX, USA.

10. Dinar M., Shah J. J., Langley P., Campana E., and Hunt G. R., 2011, "Towards a Formal Representation Model of Problem Formulation in Design," *Proceedings of ASME DETC*, Washington D.C., USA.
11. Dinar M., Shah J. J., Langley P., Hunt G. R., and Campana E., 2011, "A Structure for Representing Problem Formulation in Design," *Proceedings of the International Conference on Engineering Design*, Copenhagen, Denmark.

REFERENCES

1. Harfield S (2007) On design “problematization”: Theorising differences in designed outcomes. *Des Stud* 28:159–173
2. Dinar M, Shah JJ, Cagan J, Leifer L, Linsey JS, Smith SM, Hernandez NV (2015) Empirical Studies of Designer Thinking: Past, Present, and Future. *J Mech Des* 137:021101
3. Maher M Lou, Poon J, Boulanger S (1996) Formalising Design Exploration as Co-Evolution: A Combined Gene Approach. In: Gero JS, Sudweeks F (eds) *Adv. Form. Des. Methods CAD Proc. IFIP WG5.2 Work. Form. Des. Methods Comput. Des.* June 1995. Springer US, pp 3–30
4. Dorst K, Cross N (2001) Creativity in the design process: co-evolution of problem–solution. *Des Stud* 22:425–437
5. Amabile T (1996) Creativity in context: update to the social psychology of creativity. 317
6. Boden MA (2004) The creative mind: myths and mechanisms. 2:344
7. Newell A, Simon HA (1972) *Human Problem Solving*. Prentice-Hall, Upper Saddle River, NJ
8. Chandrasekaran B (1990) Design Problem Solving: A Task Analysis. *AI Mag* 11:59–71
9. Goel V, Pirolli P (1992) The structure of Design Problem Spaces. *Cogn Sci* 16:395–429
10. Simon HA (1973) The structure of ill structured problems. *Artif Intell* 4:181–201
11. Dorst K (2006) Design problems and design paradoxes. *Des issues* 22:4–17
12. Valkenburg R, Dorst K (1998) The Reflective Practice of Design Teams. *Des Stud* 19:249–271
13. Coyne R (2005) Wicked problems revisited. *Des Stud* 26:5–17
14. Shah JJ (2014) Design theories and models.
15. Ullman DG, Dieterich TG, Stauffer LA (1988) A model of the mechanical design process based on empirical data. *Artif Intell Eng Des Anal Manuf* 2:33–52

16. Waldron MB, Waldron KJ (1988) A time sequence study of a complex mechanical system design. *Des Stud* 9:95–106
17. Thomas JC, Carroll JM (1979) The psychological study of design. *Des Stud* 1:5–11
18. Cross N, Cross AC (1998) Expertise in Engineering Design. *Res Eng Des* 141–149
19. Atman CJ, Chimka JR, Bursic KM, Nachtmann HL (1999) A Comparison of Freshman and Senior Engineering Design Processes. *Des Stud* 20:131–152
20. Eisentraut R (1999) Styles of problem solving and their influence on the design process. *Des Stud* 20:431–437
21. Kim MHH, Kim YSS, Lee HSS, Park J a. A (2007) An underlying cognitive aspect of design creativity: Limited Commitment Mode control strategy. *Des Stud* 28:585–604
22. Jansson DG, Smith SM (1991) Design fixation. *Des Stud* 12:3–11
23. Purcell AT, Gero JS (1996) Design and other types of fixation. *Des Stud* 17:363–383
24. Shah JJ, Smith SM, Vargas-Hernandez N (2003) Metrics for measuring ideation effectiveness. *Des Stud* 24:111–134
25. Nelson BA, Wilson JO, Rosen D, Yen J (2009) Refined metrics for measuring ideation effectiveness. *Des Stud* 30:737–743
26. Kim MJ, Maher M Lou (2008) The impact of tangible user interfaces on spatial cognition during collaborative design. *Des Stud* 29:222–253
27. Lemons G, Carberry A, Swan C, Jarvin L, Rogers C (2010) The benefits of model building in teaching engineering design. *Des Stud* 31:288–309
28. Gero JS, Jiang H, Williams CB (2013) Design cognition differences when using unstructured, partially structured, and structured concept generation creativity techniques. *Int J Des Creat Innov* 1:196–214
29. Christiaans H, Dorst K (1992) An empirical study into design thinking. *Res. Des. Thinking*, N. Roozenbg. K. ...
30. Fricke G (1999) Successful approaches in dealing with differently precise design problems. *Des Stud* 20:417–429

31. Pahl G, Beitz W (1996) *Engineering Design: A Systematic Approach*. Springer, London, UK
32. Otto KN, Wood KL (2001) Product design: techniques in reverse engineering and new product development. 1071
33. Kogure M, Akao Y (1983) Quality function deployment and CWQC in Japan. *Qual Prog* 16:25–29
34. Kruger C, Cross N (2006) Solution driven versus problem driven design: strategies and outcomes. *Des Stud* 27:527–548
35. Gero JS, Mc Neill T (1998) An approach to the analysis of design protocols. *Des Stud* 19:21–61
36. Ward TB, Patterson MJ, Sifonis CM (2004) The Role of Specificity and Abstraction in Creative Idea Generation. *Creat Res J* 16:1–9
37. Ball LJ, Ormerod TC, Morley NJ (2004) Spontaneous analogising in engineering design: a comparative analysis of experts and novices. *Des Stud* 25:495–508
38. Liikkanen LA, Perttula M (2009) Exploring problem decomposition in conceptual design among novice designers. *Des Stud* 30:38–59
39. Ho C (2001) Some phenomena of problem decomposition strategy for design thinking: differences between novices and experts. *Des Stud* 22:27–45
40. Ball LJ, St.B.T. Evans J, Dennis I, Ormerod TC (1997) Problem-solving Strategies and Expertise in Engineering Design. *Think Reason* 3:247–270
41. Cai H, Do EY-L, Zimring CM (2010) Extended linkography and distance graph in design evaluation: an empirical study of the dual effects of inspiration sources in creative design. *Des Stud* 31:146–168
42. Kavakli M, Sturt C, Gero JS (2002) The structure of concurrent cognitive actions: a case study on novice and expert designers. *Des Stud* 23:25–40
43. Ahmed S, Christensen BT (2009) An In Situ Study of Analogical Reasoning in Novice and Experienced Design Engineers. *J Mech Des* 131:111004
44. Atman CJ, Cardella ME, Turns J, Adams R (2005) Comparing freshman and senior engineering design processes: an in-depth follow-up study. *Des Stud* 26:325–357

45. Goldschmidt G (1997) Capturing indeterminism: representation in the design problem space. *Des Stud* 18:441–455
46. Goldschmidt G, Tassa D (2005) How good are good ideas? Correlates of design creativity. *Des Stud* 26:593–611
47. Kan JWT, Gero JS (2008) Acquiring information from linkography in protocol studies of designing. *Des Stud* 29:315–337
48. Gero JS (1990) Design prototypes: a knowledge representation schema for design. *AI Mag* 11:26–36
49. Gero JS, Kannengiesser U (2004) The situated function-behaviour-structure framework. *Des Stud* 25:373–391
50. Pourmohamadi M, Gero JS (2011) LINKOgrapher: An Analysis Tool to Study Design Protocols Based on FBS Coding. *Proc. Int. Conf. Eng. Des. Copenhagen, Denmark*, pp 1–10
51. Anthony L, Regli WC, John JE, Lombeyda S V. (2001) An Approach to Capturing Structure, Behavior, and Function of Artifacts in Computer-Aided Design. *J Comput Inf Sci Eng* 1:186–192
52. Gero JS, Kannengiesser U (2007) Locating Creativity in a Framework of Designing for Innovation. In: León-Rovira N (ed) *Trends Comput. Aided Innov.* Springer Boston, pp 57–66
53. Sembugamoorthy V, Chandrasekaran B (1986) Functional representation of devices and compilation of diagnostic problem-solving systems. In: J K, Riesbeck C (eds) *Exp. Mem. Reason.* Lawrence Erlbaum Associates, Hillsdale, NJ, pp 47–73
54. Chandrasekaran B (1994) Functional Representation: A Brief Historical Perspective. *Appl Artif Intell* 8:173–197
55. Umeda Y, Takeda H, Tomiyama T, Yoshikawa H (1990) Function, behaviour, and structure. In: Gero JS (ed) *Appl. Artif. Intell. Eng. V1. Computational Mechanics Publications and Springer-Verlag, Berlin, Germany*, pp 177–194
56. Umeda Y, Ishii M, Yoshioka M, Shimomura Y, Tomiyama T (1996) Supporting conceptual design based on the function-behavior-state modeler. *Artif Intell Eng Des Anal Manuf* 10:275–288

57. Umeda Y, Kondoh S, Shimomura Y, Tomiyama T (2005) Development of design methodology for upgradable products based on function–behavior–state modeling. *Artif Intell Eng Des Anal Manuf* 19:161–182
58. Goel AK, Rugaber S, Vattam S (2009) Structure , Behavior and Function of Complex Systems: The Structure, Behavior, and Function Modeling Language. *Artif Intell Eng Des Anal Manuf* 23:23–35
59. Helms M, Goel AK (2014) The Four-Box Method: Problem Formulation and Analogy Evaluation in Biologically Inspired Design. *J Mech Des* 136:111106
60. Wölkl S, Shea K (2009) A computational product model for conceptual design using SysML. *Proc. ASME IDETC/CIE*
61. Larkin JH, Simon HA (1987) Why a Diagram is (Sometimes) Worth Ten Thousand Words. *Cogn Sci* 11:65–100
62. Doumont J-L (2002) Verbal versus visual: A word is worth a thousand pictures, too. *Tech Commun* 49:219–224
63. Willows DM (1978) A picture is not always worth a thousand words: Pictures as distractors in reading. *J Educ Psychol* 70:255–262
64. Miller GA (1995) WordNet: a lexical database for English. *Commun ACM* 38:39–41
65. Bohm MR, Stone RB, Simpson TW, Steva ED (2008) Introduction of a data schema to support a design repository. *Comput Des* 40:801–811
66. Uschold M (1998) Knowledge level modelling: concepts and terminology. *Knowl Eng Rev* 13:5–29
67. Sim SK, Duffy AHB (2003) Towards an ontology of generic engineering design activities. *Res Eng Des* 14:200–223
68. Srinivasan V, Chakrabarti A (2009) SAPPHERE – AN APPROACH TO ANALYSIS AND SYNTHESIS. *Proc. Int. Conf. Eng. Des.* pp 417–428
69. Srinivasan V, Chakrabarti A, Lindemann U (2013) Towards an Ontology of Engineering Design Using SAPPHERE Model. In: Chakrabarti A (ed) *CIRP Des. 2012 SE - 3*. Springer London, pp 17–26
70. Hirtz J, Stone RB, Mcadams DA, Szykman S, Wood KL (2002) A functional basis for engineering design: Reconciling and evolving previous efforts. *Res Eng Des* 13:65–82

71. Novak JD, Gowin DB (1984) Learning how to learn. Cambridge Univ Pr
72. Oxman R (2004) Think-maps: teaching design thinking in design education. *Des Stud* 25:63–91
73. Novak JD, Cañas AJ (2008) The Theory Underlying Concept Maps and How to Construct and Use Them.
74. Quillian R (1966) Semantic Memory. Carnegie Institute of Technology
75. Lehmann F (1992) Semantic networks. *Comput Math with Appl* 23:1–50
76. Hao J-X, Chi-Wai Kwok R, Yiu-Keung Lau R, Yan Yu A (2010) Predicting problem-solving performance with concept maps: An information-theoretic approach. *Decis Support Syst* 48:613–621
77. Dinar M, Shah JJ, Langley P, Hunt GR, Campana E (2011) A Structure for Representing Problem Formulation in Design. In: Culley SJ, Hicks BJ, McAlloone TC, Howard TJ, Chen W (eds) *Proc. Int. Conf. Eng. Des. Copenhagen, Denmark*, pp 392–401
78. Gero JS, Tham KW, Lee HS (1991) Behaviour: A Link Between Function and Structure in Design. *IFIP WG 5.2 Work. Conf. Intell. Comput. Aided Des.*
79. Baya V, Leifer LJ (1996) Understanding Information Management in Conceptual Design. In: Cross N, Christiaans H, Dorst K (eds) *Anal. Des. Act. John Wiley & Sons*, pp 151–168
80. Goel AK (1997) Design, analogy, and creativity. *IEEE Expert* 12:62–70
81. Dinar M, Shah JJ, Langley P, Campana E, Hunt GR (2011) Towards a Formal Representation Model of Problem Formulation in Design. *Proc. ASME IDETC/CIE*
82. Ulrich KT, Eppinger SD (2004) Product design and development. McGraw-Hill, Boston, MA, USA
83. Dieter GE, Schmidt LC (2008) Engineering Design, 4th ed. McGraw-Hill
84. Norman DA (1990) The design of everyday things. 1 Doublyay/:257
85. Simon HA (1969) The Sciences of the Artificial. MIT Press, Cambridge, MA

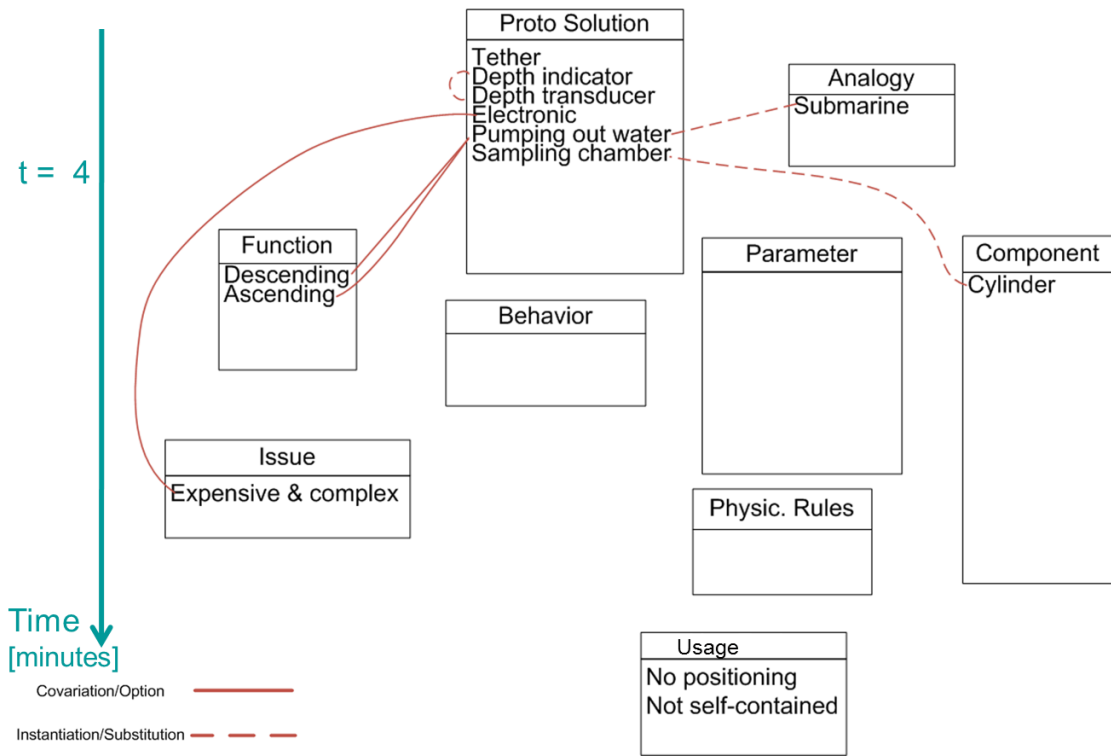
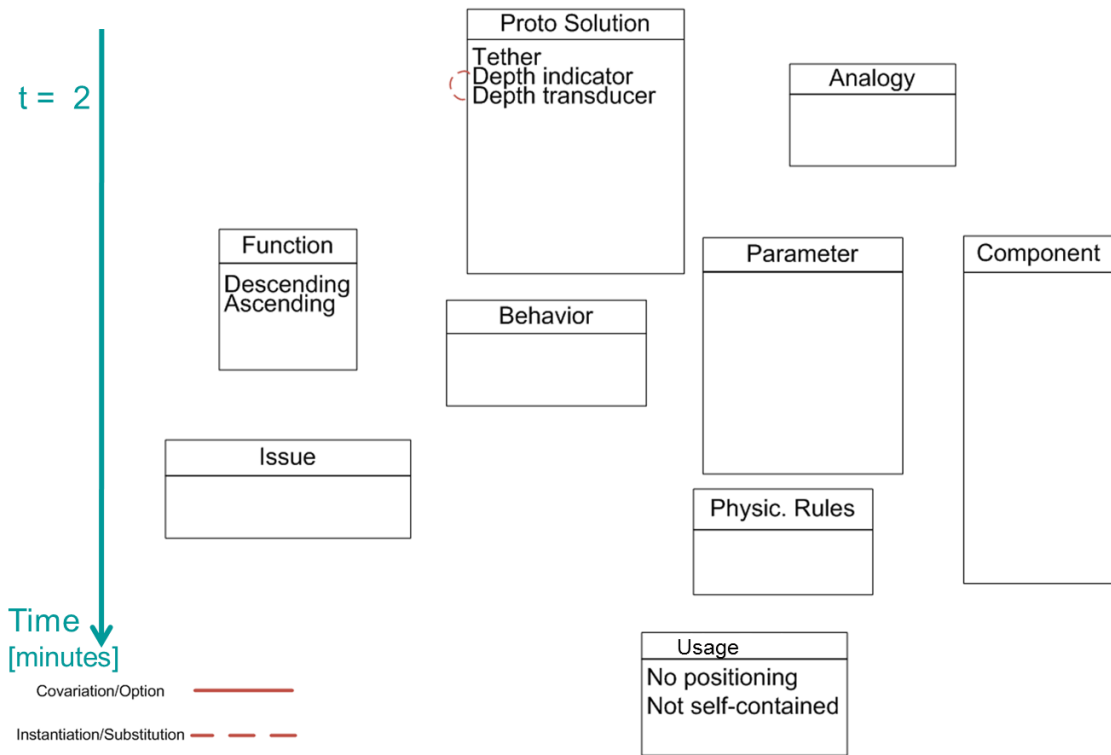
86. Dinar M, Danieleescu A, MacLellan C, Shah J, Langley P (2015) Problem Map: An ontological framework for a computational study of problem formulation in engineering design. *J Comput Inf Sci Eng* 15:1–10
87. Gero JS, Fujii H (2000) A computational framework for concept formation for a situated design agent. *Knowledge-Based Syst* 13:361–368
88. Stone RB, Wood KL (2000) Development of a functional basis for design. *J Mech Des* 122:359–370
89. Altshuller G (1998) *40 Principles: TRIZ keys to technical innovation*, 1st ed. 141
90. Pelham B, Blanton H (2007) *Conducting research in psychology: Measuring the weight of smoke*, 3rd ed. Thompson Wadsworth, Belmont, CA, USA
91. Cohen J (1960) A Coefficient of Agreement for Nominal Scales. *Educ Psychol Meas* 20:37–46
92. Fleiss JL (1971) Measuring nominal scale agreement among many raters. *Psychol Bull* 76:378
93. Gwet KL (2008) Variance Estimation of Nominal-Scale Inter-Rater Reliability with Random Selection of Raters. *Psychometrika* 73:407–430
94. Fleiss JL, Levin B, Paik MC (2003) *Statistical methods for rates and proportions*, 3rd ed. John Wiley & Sons, Hoboken, NJ, USA
95. Landis JR, Koch GG (1977) The measurement of observer agreement for categorical data. *Biometrics* 33:159–174
96. Maclellan CJ, Langley P, Shah JJ, Dinar M (2013) A Computational Aid for Problem Formulation in Early Conceptual Design. *J Comput Inf Sci Eng* 13:031005
97. Nagel RL, Perry KL, Stone RB, McAdams DA (2009) FunctionCAD: A Functional Modeling Application Based on the Function Design Framework. *Proc. ASME DETC*. ASME, San Diego, CA, USA, pp 591–600
98. Vattam S, Wiltgen B, Helms M, Goel AK, Yen J (2011) DANE: fostering creativity in and through biologically inspired design. *Des. Creat.* 2010. Springer, pp 115–122
99. Shah JJ, Millsap RE, Woodward J, Smith SM (2012) Applied Tests of Design Skills—Part 1: Divergent Thinking. *J Mech Des* 134:021005

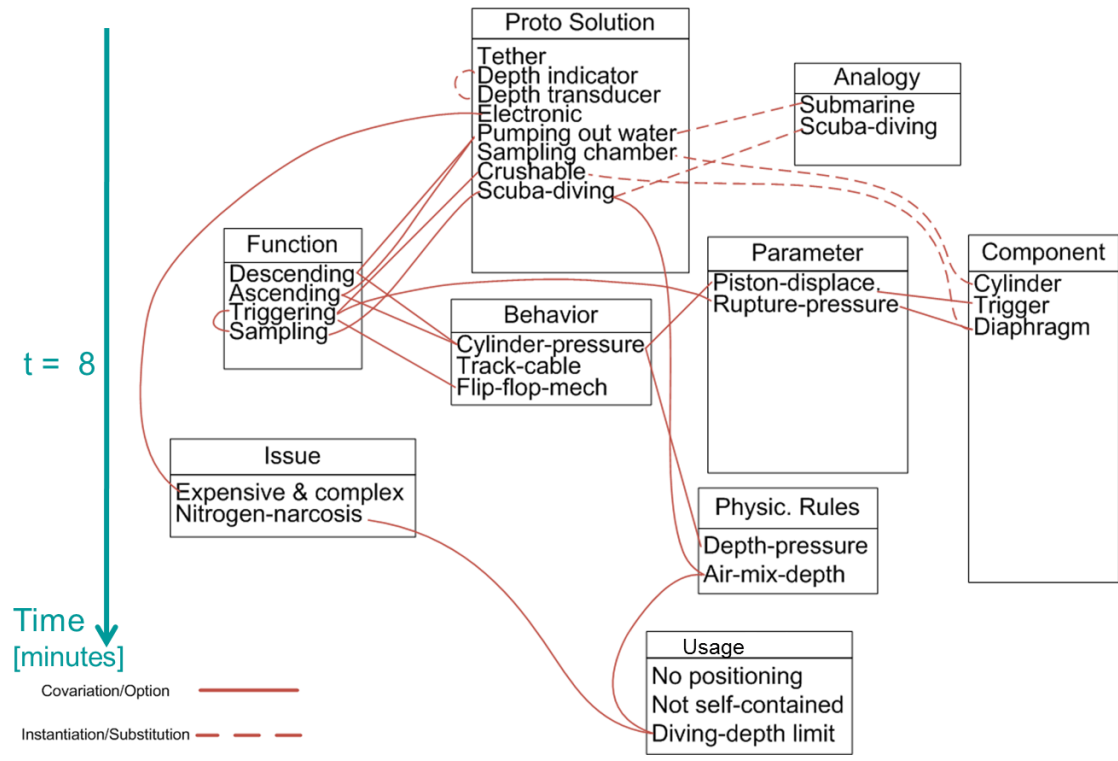
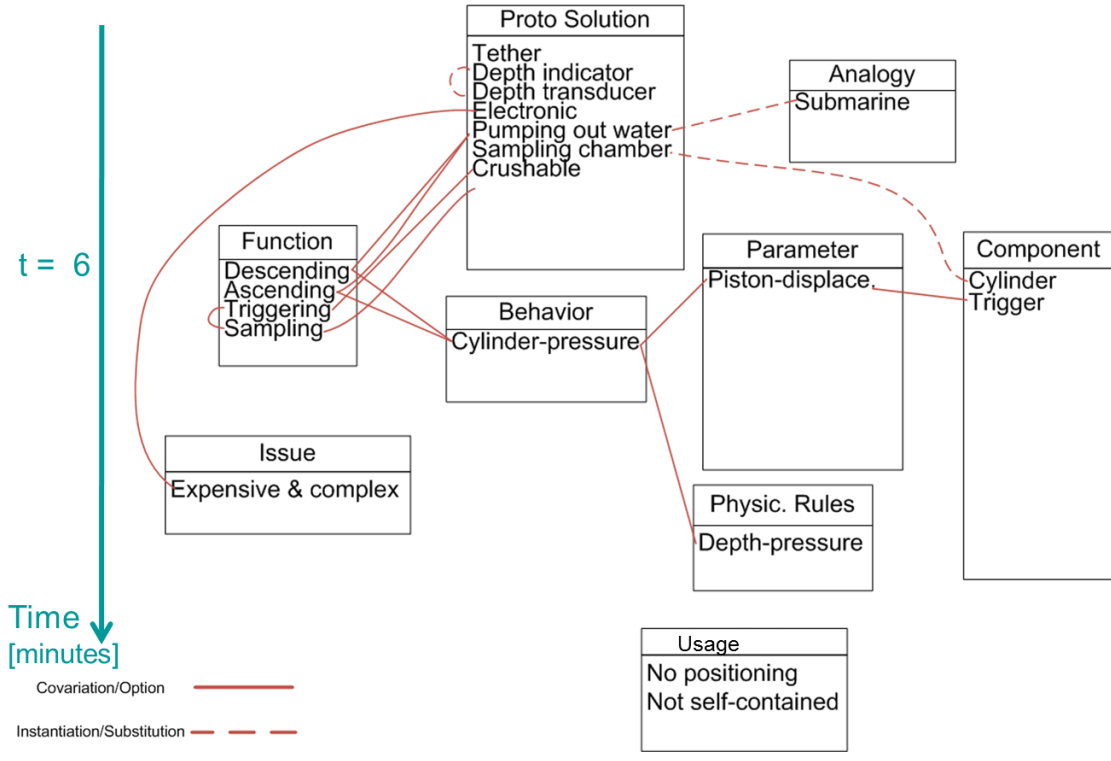
100. Dixon J, Duffey M, Irani R, Meunier K, Orelup M (1988) A proposed taxonomy of mechanical design problems. Proc. ASME Comput. Eng. Conf. San Francisco, pp 41–46
101. Gelfond M (2008) Answer Sets. In: Frank van Harmelen VL and BPBT-F of AI (ed) Handb. Knowl. Represent. Elsevier, pp 285–316
102. Gebser M, Kaminski R, Kaufmann B, Schaub T (2012) Answer Set Solving in Practice. Synth Lect Artif Intell Mach Learn 6:238
103. Gebser M, Kaminski R, Kaufmann B, Ostrowski M, Schaub T, Schneider M (2011) Potassco: The Potsdam Answer Set Solving Collection. AI Commun 24:107–124
104. Csikszentmihalyi M (1996) Creativity: flow and the psychology of discovery and invention. 1:456
105. Ward TB, Smith SM, Vaid J (1997) Creative thought: an investigation of conceptual structures and processes. 1:567
106. Montgomery DC, Runger G (2007) Applied statistics and probability for engineers, 4th ed. Wiley, Hoboken, NJ, USA
107. Tan P-N, Steinbach M, Kumar V (2005) Introduction to data mining. 1:769
108. Danieleescu A, Dinar M, Maclellan CJ, Shah JJ, Langley P (2012) The Structure of Creative Design: What Problem Maps Can Tell Us about Problem Formulation and Creative Designers. Proc. ASME IDETC/CIE
109. Quinlan JR (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA
110. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten Ian H (2011) The WEKA Data Mining Software: An Update. SIGKDD Explor 11:10–18
111. Koestler A (1964) The act of creation. Hutchinson & Co, London, UK
112. Shah JJ (2005) Identification, Measurement and Development of Design Skills in Engineering Education. In: Samuel A, Lewis W (eds) Proc. 15th Int. Conf. Eng. Des. Melbourne, Australia, p DS35_557.1
113. Shah JJ, Woodward J, Smith SM (2013) Applied Tests of Design Skills—Part II: Visual Thinking. J Mech Des 135:71004

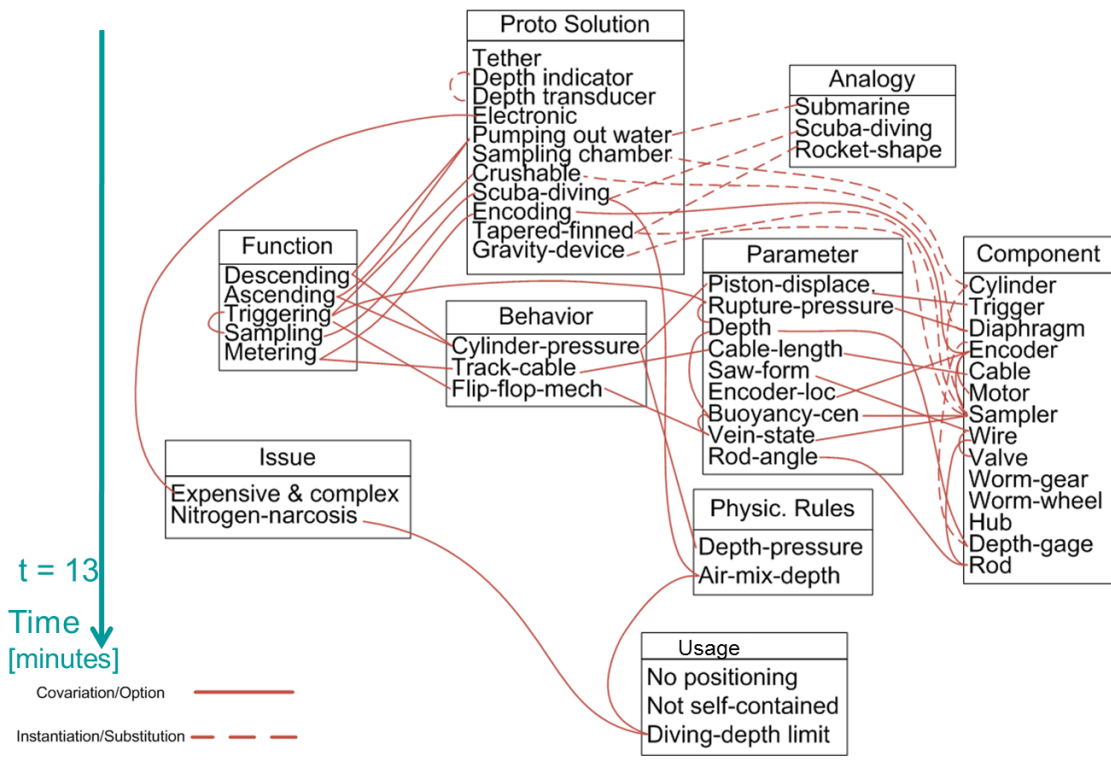
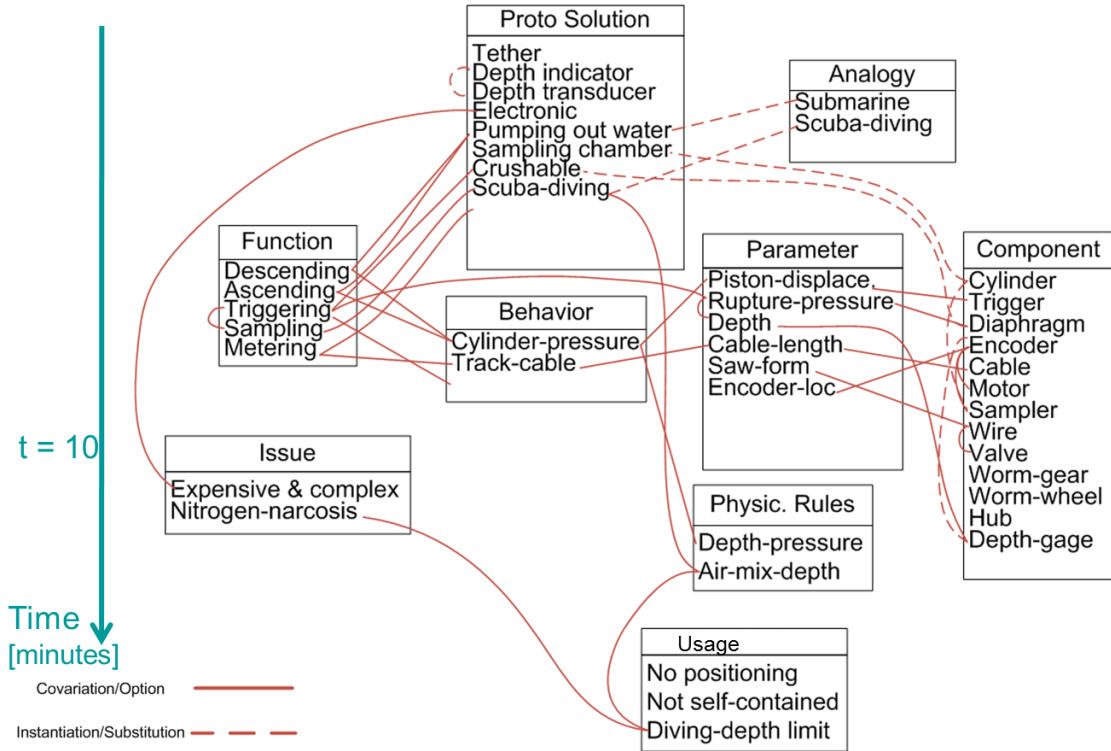
114. Khorshidi M, Woodward J, Shah JJ (2012) Towards a Comprehensive Test of Qualitative Reasoning Skill in Design. Proc. ASME IDETC/CIE. ASME, Chicago, IL, USA, p 889
115. Khorshidi M, Shah JJ, Woodward J (2013) Rethinking the Comprehensive Test on Qualitative Reasoning for Designers. Proc. ASME IDETC/CIE. American Society of Mechanical Engineers, Portland, OR, p V005T06A027
116. Kano N, Seraku N, Takahashi F, Tsuji S (1984) Attractive quality and must be quality. Quality 14:39–48
117. Dinar M, Shah JJ, Todeti SR (2015) Towards a Comprehensive Test of Problem Formulation Skill in Design. In: Chakrabarti A, Taura T, Nagai Y (eds) Proc. Third Int. Conf. Des. Creat. Bangalore, India, pp 19–26
118. Khorshidi M, Shah JJ, Woodward J (2014) Applied Tests of Design Skills—Part III: Abstract Reasoning. J Mech Des 136:101101
119. Cross N, Christiaans H, Dorst K (1996) Analysing design activity. 463
120. Muggleton S, de Raedt L (1994) Inductive Logic Programming: Theory and methods. J Log Program 19-20:629–679

APPENDIX A

DESIGNERS' FORMULATION SHOWN IN SNAPSHOTS







APPENDIX B

ASP ENCODINGS OF STRATEGIES

```
abstraction_strategy = "strategy(abstraction,Ent_parent):-  
entity(Ent_parent,Desc_parent,T_parent), entity(Ent_child,Desc_child,T_child),  
parent_of(Ent_parent,Ent_child,T_parent_of), T_parent>T_child."
```

```
entity_depth_prevalence_strategy = "strategy(entity_depth_prevalence,Ent_parent):-  
parent_of(Ent_parent, Ent_child,T_parent_of),  
interrelate_bi_directionally(Ent_parent,Any,T_interrelate), T_interrelate>T_parent_of,  
not violate_edp_strategy(Ent_parent). violate_edp_strategy(Ent_parent):-  
parent_of(Ent_parent,Ent_child,T_parent_of),  
interrelate_bi_directionally(Ent_parent,Any,T_interrelate), T_interrelate<T_parent_of."
```

```
order_req_use_strategy = "strategy(order_req_use,Requirement):-  
requirement(Requirement,Desc_req,T_req), usescenario(Usescenario,Desc_use,T_use),  
function(Function,Desc_fun,T_fun), artifact(Artifact,Desc_art,T_art),  
behavior(Behavior,Desc_beh,T_beh),  
interrelate_bi_directionally(Usescenario,Requirement,T_use_req),  
interrelate_bi_directionally(Function,Requirement,T_fun_req),  
interrelate_bi_directionally(Artifact,Requirement,T_art_req),  
interrelate_bi_directionally(Behavior,Requirement,T_beh_req), T_fun_req>T_use_req,  
T_art_req>T_use_req, T_beh_req>T_use_req.\n\  
strategy(order_req_use,Requirement):- requirement(Requirement,Desc_req,T_req),  
usescenario(Usescenario,Desc_use,T_use), function(Function,Desc_fun,T_fun),  
behavior(Behavior,Desc_beh,T_beh), issue(Issue,Desc_iss,T_iss),
```

```

interrelate_bi_directionally(Usescenario,Requirement,T_use_req),
interrelate_bi_directionally(Function,Requirement,T_fun_req),
interrelate_bi_directionally(Behavior,Requirement,T_beh_req),
interrelate_bi_directionally(Issue,Requirement,T_iss_req), T_fun_req>T_use_req,
T_beh_req>T_use_req, T_iss_req>T_use_req.\n\
    strategy(order_req_use,Requirement):- requirement(Requirement,Desc_req,T_req),
usescenario(Usescenario,Desc_use,T_use), artifact(Artifact,Desc_art,T_art),
behavior(Behavior,Desc_beh,T_beh), issue(Issue,Desc_iss,T_iss),
interrelate_bi_directionally(Usescenario,Requirement,T_use_req),
interrelate_bi_directionally(Artifact,Requirement,T_art_req),
interrelate_bi_directionally(Behavior,Requirement,T_beh_req),
interrelate_bi_directionally(Issue,Requirement,T_iss_req), T_art_req>T_use_req,
T_beh_req>T_use_req, T_iss_req>T_use_req.\n\
    strategy(order_req_use,Requirement):- requirement(Requirement,Desc_req,T_req),
usescenario(Usescenario,Desc_use,T_use), function(Function,Desc_fun,T_fun),
artifact(Artifact,Desc_art,T_art),
interrelate_bi_directionally(Usescenario,Requirement,T_use_req),
interrelate_bi_directionally(Function,Requirement,T_fun_req),
interrelate_bi_directionally(Artifact,Requirement,T_art_req), T_fun_req>T_use_req,
T_art_req>T_use_req.\n\
    strategy(order_req_use,Requirement):- requirement(Requirement,Desc_req,T_req),
usescenario(Usescenario,Desc_use,T_use), function(Function,Desc_fun,T_fun),
behavior(Behavior,Desc_beh,T_beh),

```

interrelate_bi_directionally(Usescenario,Requirement,T_use_req),
interrelate_bi_directionally(Function,Requirement,T_fun_req),
interrelate_bi_directionally(Behavior,Requirement,T_beh_req), T_fun_req>T_use_req,
T_beh_req>T_use_req.\n\

strategy(order_req_use,Requirement):- requirement(Requirement,Desc_req,T_req),
usescenario(Usescenario,Desc_use,T_use), function(Function,Desc_fun,T_fun),
issue(Issue,Desc_iss,T_iss),

interrelate_bi_directionally(Usescenario,Requirement,T_use_req),
interrelate_bi_directionally(Function,Requirement,T_fun_req),
interrelate_bi_directionally(Issue,Requirement,T_iss_req), T_fun_req>T_use_req,
T_iss_req>T_use_req.\n\

strategy(order_req_use,Requirement):- requirement(Requirement,Desc_req,T_req),
usescenario(Usescenario,Desc_use,T_use), artifact(Artifact,Desc_art,T_art),
behavior(Behavior,Desc_beh,T_beh),

interrelate_bi_directionally(Usescenario,Requirement,T_use_req),
interrelate_bi_directionally(Artifact,Requirement,T_art_req),
interrelate_bi_directionally(Behavior,Requirement,T_beh_req), T_art_req>T_use_req,
T_beh_req>T_use_req.\n\

strategy(order_req_use,Requirement):- requirement(Requirement,Desc_req,T_req),
usescenario(Usescenario,Desc_use,T_use), artifact(Artifact,Desc_art,T_art),
issue(Issue,Desc_iss,T_iss),

interrelate_bi_directionally(Usescenario,Requirement,T_use_req),
interrelate_bi_directionally(Artifact,Requirement,T_art_req),


```

interrelate_bi_directionally(Issue,Requirement,T_iss_req), T_art_req>T_use_req,
T_iss_req>T_use_req.\n\
    strategy(order_req_use,Requirement):- requirement(Requirement,Desc_req,T_req),
usesenario(Usesenario,Desc_use,T_use), behavior(Behavior,Desc_beh,T_beh),
issue(Issue,Desc_iss,T_iss),
interrelate_bi_directionally(Usesenario,Requirement,T_use_req),
interrelate_bi_directionally(Behavior,Requirement,T_beh_req),
interrelate_bi_directionally(Issue,Requirement,T_iss_req), T_beh_req>T_use_req,
T_iss_req>T_use_req."

```

```

    order_req_fun_strategy = "strategy(order_req_fun,Requirement):-
requirement(Requirement,Desc_req,T_req), function(Function,Desc_fun,T_fun),
artifact(Artifact,Desc_art,T_art), behavior(Behavior,Desc_beh,T_beh),
issue(Issue,Desc_iss,T_iss),
interrelate_bi_directionally(Function,Requirement,T_fun_req),
interrelate_bi_directionally(Artifact,Requirement,T_art_req),
interrelate_bi_directionally(Behavior,Requirement,T_beh_req),
interrelate_bi_directionally(Issue,Requirement,T_iss_req), T_art_req>T_fun_req,
T_beh_req>T_fun_req, T_iss_req>T_fun_req.\n\
    strategy(order_req_fun,Requirement):- requirement(Requirement,Desc_req,T_req),
function(Function,Desc_fun,T_fun), artifact(Artifact,Desc_art,T_art),
behavior(Behavior,Desc_beh,T_beh),
interrelate_bi_directionally(Function,Requirement,T_fun_req),

```

```

interrelate_bi_directionally(Artifact,Requirement,T_art_req),
interrelate_bi_directionally(Behavior,Requirement,T_beh_req), T_art_req>T_fun_req,
T_beh_req>T_fun_req.\n\
    strategy(order_req_fun,Requirement):- requirement(Requirement,Desc_req,T_req),
function(Function,Desc_fun,T_fun), artifact(Artifact,Desc_art,T_art),
issue(Issue,Desc_iss,T_iss),
interrelate_bi_directionally(Function,Requirement,T_fun_req),
interrelate_bi_directionally(Artifact,Requirement,T_art_req),
interrelate_bi_directionally(Issue,Requirement,T_iss_req), T_art_req>T_fun_req,
T_iss_req>T_fun_req.\n\
    strategy(order_req_fun,Requirement):- requirement(Requirement,Desc_req,T_req),
function(Function,Desc_fun,T_fun), behavior(Behavior,Desc_beh,T_beh),
issue(Issue,Desc_iss,T_iss),
interrelate_bi_directionally(Function,Requirement,T_fun_req),
interrelate_bi_directionally(Behavior,Requirement,T_beh_req),
interrelate_bi_directionally(Issue,Requirement,T_iss_req), T_beh_req>T_fun_req,
T_iss_req>T_fun_req.\n\
    strategy(order_req_fun,Requirement):- requirement(Requirement,Desc_req,T_req),
function(Function,Desc_fun,T_fun), artifact(Artifact,Desc_art,T_art),
behavior(Behavior,Desc_beh,T_beh), issue(Issue,Desc_iss,T_iss),
interrelate_bi_directionally(Function,Requirement,T_fun_req),
interrelate_bi_directionally(Artifact,Requirement,T_art_req), T_art_req>T_fun_req.\n\

```

```

strategy(order_req_fun,Requirement):- requirement(Requirement,Desc_req,T_req),
function(Function,Desc_fun,T_fun), artifact(Artifact,Desc_art,T_art),
behavior(Behavior,Desc_beh,T_beh), issue(Issue,Desc_iss,T_iss),
interrelate_bi_directionally(Function,Requirement,T_fun_req),
interrelate_bi_directionally(Behavior,Requirement,T_beh_req),
T_beh_req>T_fun_req.\n\

```

```

strategy(order_req_fun,Requirement):- requirement(Requirement,Desc_req,T_req),
function(Function,Desc_fun,T_fun), artifact(Artifact,Desc_art,T_art),
behavior(Behavior,Desc_beh,T_beh), issue(Issue,Desc_iss,T_iss),
interrelate_bi_directionally(Function,Requirement,T_fun_req),
interrelate_bi_directionally(Issue,Requirement,T_iss_req), T_iss_req>T_fun_req."

```

```

forward_order_strategy = "strategy(forward_order,Requirement):-
requirement(Requirement,Desc_req,T_req), usescenario(Usescenario,Desc_use,T_use),
function(Function,Desc_fun,T_fun), artifact(Artifact,Desc_art,T_art),
behavior(Behavior,Desc_beh,T_beh), issue(Issue,Desc_iss,T_iss),
interrelate_bi_directionally(Usescenario,Requirement,T_use_req),
interrelate_bi_directionally(Function,Requirement,T_fun_req),
interrelate_bi_directionally(Artifact,Requirement,T_art_req),
interrelate_bi_directionally(Behavior,Requirement,T_beh_req),
interrelate_bi_directionally(Issue,Requirement,T_iss_req), T_fun_req>T_use_req,
T_art_req>T_use_req, T_beh_req>T_use_req, T_iss_req>T_use_req."

```

```
requirement_depth_prevalence_strategy =  
"strategy(requirement_depth_prevalence,Ent_parent):- requirement(Ent_parent,Desc,T),  
parent_of(Ent_parent,Ent_child,T_parent_of),  
interrelate_bi_directionally(Ent_parent,Any,T_interrelate), T_interrelate>T_parent_of,  
not violate_edp_strategy(Ent_parent). violate_edp_strategy(Ent_parent):-  
parent_of(Ent_parent,Ent_child,T_parent_of),  
interrelate_bi_directionally(Ent_parent,Any,T_interrelate), T_interrelate<T_parent_of."
```

```
usescenario_depth_prevalence_strategy =  
"strategy(usescenario_depth_prevalence,Ent_parent):- usescenario(Ent_parent,Desc,T),  
parent_of(Ent_parent,Ent_child,T_parent_of),  
interrelate_bi_directionally(Ent_parent,Any,T_interrelate), T_interrelate>T_parent_of,  
not violate_edp_strategy(Ent_parent). violate_edp_strategy(Ent_parent):-  
parent_of(Ent_parent,Ent_child,T_parent_of),  
interrelate_bi_directionally(Ent_parent,Any,T_interrelate), T_interrelate<T_parent_of."
```

```
function_depth_prevalence_strategy =  
"strategy(function_depth_prevalence,Ent_parent):- function(Ent_parent,Desc,T),  
parent_of(Ent_parent,Ent_child,T_parent_of),  
interrelate_bi_directionally(Ent_parent,Any,T_interrelate), T_interrelate>T_parent_of,  
not violate_edp_strategy(Ent_parent). violate_edp_strategy(Ent_parent):-  
parent_of(Ent_parent,Ent_child,T_parent_of),  
interrelate_bi_directionally(Ent_parent,Any,T_interrelate), T_interrelate<T_parent_of."
```

```

artifact_depth_prevalence_strategy =
"strategy(artifact_depth_prevalence,Ent_parent):- artifact(Ent_parent,Desc,T),
parent_of(Ent_parent,Ent_child,T_parent_of),
interrelate_bi_directionally(Ent_parent,Any,T_interrelate), T_interrelate>T_parent_of,
not violate_edp_strategy(Ent_parent). violate_edp_strategy(Ent_parent):-
parent_of(Ent_parent,Ent_child,T_parent_of),
interrelate_bi_directionally(Ent_parent,Any,T_interrelate), T_interrelate<T_parent_of."

```

```

behavior_depth_prevalence_strategy =
"strategy(behavior_depth_prevalence,Ent_parent):- behavior(Ent_parent,Desc,T),
parent_of(Ent_parent,Ent_child,T_parent_of),
interrelate_bi_directionally(Ent_parent,Any,T_interrelate), T_interrelate>T_parent_of,
not violate_edp_strategy(Ent_parent). violate_edp_strategy(Ent_parent):-
parent_of(Ent_parent,Ent_child,T_parent_of),
interrelate_bi_directionally(Ent_parent,Any,T_interrelate), T_interrelate<T_parent_of."

```

```

conflict_identification_strategy = "strategy(conflict_issue,Issue):-
issue(Issue,Desc_issue,T_issue), interrelate_bi_directionally(Issue,Req_1,T1),
interrelate_bi_directionally(Issue,Req_2,T2), requirement(Req_1,Desc_req1,T_req1),
requirement(Req_2,Desc_req2,T_req2),Desc_req1!=Desc_req2."

```

```

    problem_driven_approach_strategy = "violate strategy(problem_driven_approach):-
requirement(Requirement,Desc_req,T_requirement),
usescenario(Usescenario,Desc_use,T_usescenario),
function(Function,Desc_fn,T_function), artifact(Artifact,Desc_art,T_artifact),
behavior(Behavior,Desc_beh,T_behavior),
use_req(Usescenario,Requirement,T_use_req),
fun_req(Function,Requirement,T_fun_req), art_req(Artifact,Requirement,T_art_req),
beh_req(Behavior,Requirement,T_beh_req),"

```

```

    coevolutionary_requirement_elicitation_strategy =
"violate_cre_strategy(Requirement,Artifact,Any):-
requirement(Requirement,Desc_req,T_req), artifact(Artifact,Desc_art,T_art),
interrelate_bi_directionally(Requirement,Artifact,T_req_art),
entity(Any,Desc_ent,T_ent), T_req>T_art, T_ent>T_art, T_ent<T_req_art,
Any!=Artifact, Any!=Requirement. \n\

```

```

    not_strategy(coevol_req_elicitation,Requirement,Artifact):-
requirement(Requirement,Desc_req,T_req),
artifact(Artifact,Desc_art,T_art),interrelate_bi_directionally(Requirement,Artifact,T_req_
art), entity(Any,Desc_ent,T_ent), T_req>=T_art,
violate_cre_strategy(Requirement,Artifact,Any). \n\

```

```

    strategy(coevol_req_elicitation,Requirement,Artifact) :- not
not_strategy(coevol_req_elicitation,Requirement,Artifact),requirement(Requirement,Des
c_req,T_req), artifact(Artifact,Desc_art,T_art)."

```

```

    coevolutionary_general_requiremenet_elicitation_strategy =
"violate_cre_strategy(Requirement,Entity,Any):-
requirement(Requirement,Desc_req,T_req), entity(Entity,Desc_ent,T_ent),
interrelate_bi_directionally(Requirement,Entity,T_req_ent),
entity(Any,Desc_any,T_any), T_req>T_ent, T_any>T_ent, T_any<T_req_ent,
Any!=Entity, Any!=Requirement. \n\

    not_strategy(coevol_req_elicitation,Requirement,Entity):-
requirement(Requirement,Desc_req,T_req), entity(Entity,Desc_ent,T_ent),
interrelate_bi_directionally(Requirement,Entity,T_req_ent),
entity(Any,Desc_any,T_any), T_req>=T_ent,
violate_cre_strategy(Requirement,Entity,Any). \n\

    strategy(coevol_req_elicitation,Requirement,Entity) :- not
not_strategy(coevol_req_elicitation,Requirement,Entity),requirement(Requirement,Desc_
req,T_req), entity(Entity,Desc_ent,T_ent)."

```

APPENDIX C

EXCERPTS OF A CODED PROTOCOL (DP_1)

Code	Data
solutionPrinciple(sl_device)	Mechanical device,...
requirement(rq_freshwater_sample), requirementType(rq_freshwater_sample, given)	...fresh water samples.
requirement(rq_max_depth_500meters), requirementType(rq_max_depth_500meters , given)	Let's see, to 500 meter depth. Okay, hmm.
requirement(rq_not_attached), requirementType(rq_not_attached, given)	Device must not be attached to the boat and must be ...
requirement(rq_known_depth), requirementType(rq_known_depth, given), requirement(rq_depth_accuracy_10meters), requirementType(rq_depth_accuracy_10met e rs, given)	...within 10 meters of pre-adjusted depth
requirement(rq_sample_size_.5liters), requirementType(rq_sample_size_.5liters, given)	I think that's coming from out there. And return with a point five liter sample of water from that depth.
requirement(rq_mechanical_device), requirementType(rq_mechanical_device, given), requirement(rq_mechanical_only)	Umm so, well, I'll attempt to answer your – so mechanical only or does it – it can be electrical or there's I guess you can't answer that. But, hmm. It says mechanical so it implies mechanical only device. Okay. It's just that in Larry's thing he said electromechanical but okay, mechanical.

Code	Data
<p>physicalEffect(ph_pressure), issue(iu_stop_at_known_depth, sl_device, "I'm trying to think of how to get it to go down and stop at a certain depth")</p>	<p>All right, let's see. Kind of what I'm thinking is how would this work, is, ah, you somehow let's – how would that work? Set, obviously setting it to go to a certain depth, hmm, a certain pressure, um, I'm trying to think of how to get it to go down and stop at a certain depth.</p>
<p>goal(gl_do_not_bouce_off_bottom_of_lake)</p>	<p>So let's see, pretty much need to work with the water pressure, we can't be attached. Yeah, we're not going to like bounce off the bottom of the lake and come up some amount.</p>
<p>delete(gl_do_not_bounce_off_bottom_of_la k e)</p>	<p>Well, I suppose you could go all the way down and then come back up.</p>
<p>solutionPrinciple(sl_pressure_activated)</p>	<p>Um let's see. I'm just jotting here, let's see, so it's going to be like pressure-activated.</p>
<p>function(fn_collect_sample), parameter(pr_known_depth), parameterFunction(pr_known_depth, fn_collect_sample)</p>	<p>And it – so it needs to be able to open up and accept a sample when it gets to a certain depth...</p>
<p>function(fn_make_buoyant), function(fn_ascend)</p>	<p>...and then once it gets the sample it needs to do something to make it buoyant and come back to the surface</p>
<p>goal(gl_do_not_bouce_off_bottom_of_lake)</p>	<p>So I'll just say pressure activated to accept sample, hmm. Then it must become buoyant. Buoyant to return and</p>

Code	Data
	oh, I see, to known depths down to maximum of 500, so you're probably not going to want to bounce it off the bottom because the bottom could be lower than that.
solutionPrinciple(sl_transistor)	Uh, let's see. Hmm. What am I thinking? I'm thinking something like a, what do I need to do that? Some kind of like a transistor kind of thing...
solutionPrinciple(sl_diaphragm), physicalEffect(ph_force)	...or a diaphragm so you're using a smaller amount of pressure to move pressure over a diaphragm to have enough force to do something.
solutionPrinciple(sl_compressed_air)	Um, I think I pretty much need to, in order to make it buoyant again, I pretty much – I think I'm going to – I don't know for sure. I need some kind of like compressed air on board.
function(fn_expand_vessel), functionObject(fn_expand_vessel, sl_device)	Compressed air question mark. Um, either that or does it work for it to, for my um container, my vessel to just get bigger when it wants to come up? Just kind of expand bigger...
function(fn_pull_vacuum), functionObject(fn_pull_vacuum, sl_device)	...and pull a little bit of a vacuum on the inside.
issue(iu_hard_to_pull_vacuum, fn_pull_vacuum, "this [the vacuum] might be	That might be kind of hard to do though. Uh, yeah. Especially under all that pressure.

Code	Data
hard to do")	
equation(eq_atm_related_to_feet, one atm for like 32 feet of water, concrete)	Okay. Hm. What were we, we were just talking at – not, yeah, not about this at lunch, but a couple of my buddies – let’s see, one atmosphere’s like 32 feet of water.
function(fn_drop_in_water), functionObject(fn_drop_in_water, sl_device), physicalEmbodiment(em_door)	Um, and so yeah, we chuck this thing over the edge and then have like a little door, a door on the inside, kind of showing a side view here,...
physicalEmbodiment(em_gasket), solutionPrinciple(sl_seal), parentOf(sl_seal, em_gasket)	...got a little gasket to seal it.
physicalEmbodiment(em_screw)	And then, let’s see, I have like ah, probably something with a screw on it.
physicalEmbodiment(em_handle), connects(em_handle, em_screw), parameter(pr_depth_of_screw), parameterEmbodiment(pr_depth_of_screw, em_screw)	A little handle coming out the side, so I can set how far in I want the screw,...
physicalEmbodiment(em_compression_spring), connects(em_compression_spring, em_screw), function(fn_compress_spring), functionObject(fn_compress_spring, em_compression_spring), realizes(em_screw, fn_compress_spring)	... and then the screw is compressing a big old compression spring.

Code	Data
<pre>issue(iu_water_makes_device_sink, sl_device, "if I let water in and do nothing else, it's going to start dropping fast")</pre>	<p>And let's see. If I'm letting water in and doing nothing else, it's going to start dropping faster.</p>
<pre>function(fn_close_device), issue(iu_how_to_close_device, fn_close_device, "how the heck am I going to close the device")</pre>	<p>Let's see, and I need this thing to close back up again too, so how the heck am I going to do that? Hm.</p>
<pre>parameter(pr_delta_time_to_open), parameterFunction(pr_delta_time_to_open, fn_collect_sample), function(fn_stabilize_device_at_depth)</pre>	<p>I'm wondering if when I send this thing down, if I have it open fast enough such that it doesn't take very long to get it's half a liter – that's not that much water in there, then I don't have to be concerned about stabilizing this thing at a particular depth.</p>
<pre>before(fn_collect_sample, fn_close_device), before(fn_close_device, fn_ascend)</pre>	<p>So I don't need to stop it then. Um, pull the water in and close it up and head to the top. I can just be kind of moving as I quickly gulp in the half a liter of water. But after I gulp it, I need to close it back up. I don't think there's any way I can get away with not closing it back up.</p>
<pre>issue(iu_contamination_of_sample, fn_collect_sample, "can't get a lot of contamination from water going back up")</pre>	<p>Yeah. Let's see. I mean if I gulped it – well, no, but what I was thinking is if I bring it in slowly then launch back to the surface then I might not get much contamination from water when I get back up.</p>

Code	Data
<pre>issue(iu_do_I_need_separate_chamber_for_ c ompressed_air, sl_compressed_air, "do I need a separate chamber for this compressed air thing?")</pre>	<p>But, nah. Let's see. Well, I've got my compressed air just waiting to be deployed here. Hmm, what am I thinking here? Hmm. I'm thinking something. Um, I'm trying to decide if I need a separate chamber for this compressed air thing. I probably do.</p>
<pre>physicalEmbodiment(em_bladder), physicalEmbodiment(em, balloon)</pre>	<p>Uh, I'm thinking I'll have like a bladder, like a balloon, bladder, whatever, that will get filled up at some point.</p>
<pre>realizes(sl_compressed_air, fn_close_device), physicalEmbodiment(em_compressed_air_t a nk), parentOf(sl_compressed_air, em_compressed_air_tank)</pre>	<p>And let's see, do I want it in the same housing as my sample taker or not? Probably not. Um, but I'm also thinking I want to use that compressed air to close my little door again.</p>
<pre>function(fn_let_out_compressed_air), before(fn_let_out_compressed_air, fn_close_device), parameterFunction(pr_known_depth, fn_let_out_compressed_air)</pre>	<p>Um, hmm. Yeah, I'm pretty sure I can, I mean I can come up with something – concept here pretty quickly that will trip the compressed air cylinder to let air out, uh, when I get to the depth, when I can have it trip off my little door opening. Hmm. Or maybe I have a different idea. Hmm. Let's see. So if I go to another page, can I rip out the page or not?</p>
<pre>connects(sl_device,</pre>	<p>Okay. I'm going to rip out this and just</p>

Code	Data
em_compressed_air_tank)	set it aside here. So I'm thinking this might simplify it a little bit. So let's see, I've got my big, I've got my container, I've got my compressed air up – well, let's just say it's wherever it is, it's – it doesn't even have to – I'm just going to draw it on the outside of the containers where my compressed air tank.
physicalEmbodiment(em_open_bottomed_device, parentOf(sl_device, em_open_bottomed_device), parameter(pr_num_holes_on_sides), parameterEmbodiment(pr_num_holes_on_sides, em_open_bottomed_device)	And inside, let's see. On this one, yeah, I have like a – in this case what I'm thinking is the bottom of my container is open on the sides in a number of spots. Or – yeah, yeah, some of this will take a little work, probably. Yeah, let's just say it's open on the bottom here.
connects(em_open_bottomed_device, em_gasket)	I have a gasket here.
parameter(pr_door_open), parameterFunction(pr_door_open, fn_descend)	So in this scenario, basically the tank, the collection vessel is wide open and so water's kind of – hmm, flushing through it as I drop it down, and as I get to depth I close it up. So it's open the whole time until I get to depth and then close it up.

APPENDIX D

REGRESSORS OF STATE COUNTS MODELS

Comparison of regressors of P-map state counts for two problems; italic: same sign;
bold: P < 0.2; starred: lowest P value above 0.2

DP_5 max quality	4.00	-0.14	0.07	0.14*	0.05	0.32	0.23	0.05	-0.01	-0.11
DP_4 max quality	7.57	0.09	-0.18	-0.10	-0.03	-0.07	0.05	-0.04	-0.10	0.04
DP_5 avg. quality	3.53	0.15	-0.29	-0.02	-0.05	0.27	0.43*	-0.01	0.08	0.03
DP_4 avg. quality	5.74	-0.03	0.02	-0.05	-0.06	0.19	0.21*	-0.03	0.07	-0.01
DP_5 max novelty	4.17	-0.18	0.18	0.08	0.25	-0.40	-0.13	0.05	-0.08	-0.06
DP_4 max novelty	7.71	0.09	0.32	0.22	0.04	-2.09	0.92*	-0.07	-0.94	0.01
DP_5 avg. novelty	2.74	-0.05	-0.11	0.03	0.16	-.52*	0.04	0.05	-0.11	-0.01
DP_4 avg. novelty	5.64	-0.02	0.31	0.20	0.10	-1.08	0.60	-0.05	-0.58	0.04
DP_5 variety	3.42	-0.27	0.86	0.00	0.27	0.56	-0.42	0.08	-0.34	-0.17
DP_4 variety	5.33	0.17	0.07	0.01	0.01	-2.55	1.35	-0.03	-1.02	0.01
DP_5 quantity	1.85	0.12	0.14	-0.02	-0.08	0.59	-.54*	-0.11	0.21	-0.08
DP_4 quantity	4.72	0.30*	-0.26	-0.05	0.12	-2.26	0.62	-0.07	-0.60	-0.05
Variable	Constant	requirement	use scenario	function	artifact	behavior	issue	isolated requirement	isolated use scenario	isolated function

DP_5 max quality	0.03	-0.37	0.11	0.03	-0.07	0.13	-0.15	0.12
DP_4 max quality	0.11	0.04	-0.11	0.19	0.04	0.15	0.10	-0.36
DP_5 avg. quality	0.07	-0.40	-0.11	0.29	-0.09	0.26	0.10	-0.48
DP_4 avg. quality	0.11	-0.23	0.01	0.03	-0.01	0.05	0.14	-0.43
DP_5 max novelty	0.01	0.20	0.16	-0.30	0.04	-0.14	-0.08	0.37
DP_4 max novelty	-0.04	1.35	-0.14	-0.08	0.00	-0.40	1.01	-1.05
DP_5 avg. novelty	0.01	0.40	0.05	0.04	0.02	-0.09	-0.04	0.11
DP_4 avg. novelty	-0.09	0.59	-0.01	-0.13	-0.07	-0.36	0.74	-0.75
DP_5 variety	-0.09	-0.28	0.25	-0.95	0.16	-0.05	-0.57	0.84
DP_4 variety	0.23	1.60	-0.24	0.11	0.12	-0.39	1.12	-1.12
DP_5 quantity	0.27	-0.60	-0.18	-0.21	0.04	0.35	-0.13	0.95
DP_4 quantity	0.07	1.66	-0.37	0.20	0.18	-0.27	0.66	-0.26
Variable	isolated artifact	isolated behavior	disconnected requirement	disconnected use scenario	disconnected function	disconnected artifact	disconnected behavior	disconnected issue

APPENDIX E
REGRESSORS AFTER BACKWARD ELIMINATION

Regressors after backward elimination for the combined P-map state and strategies variables; italic: same sign in DP_4 and DP_5

DP_5 max quality	4.8	-0.61	0.94	0.31		0.58		0.16	-0.68
DP_4 max quality	7.65	0.06	-0.15	-0.02	-0.1	-0.83	0.64		-0.13
DP_5 avg. quality	4.96	-0.75	1.27	0.43	-0.35	0.69		0.13	-0.87
DP_4 avg. quality	6.47	-0.08				-0.47	0.66	0.03	
DP_5 max novelty	3.11	-0.81	1.3	0.17		0.54		0.1	-0.41
DP_4 max novelty	8.5		0.22	0.29		-3.52	2.42		-0.98
DP_5 avg. novelty	2.13	-0.38	0.3				-0.25	0.08	
DP_4 avg. novelty	5.54		0.38	0.31		-2.22	1.56		-0.62
DP_5 variety	-0.34	0.59		-1.09	0.71	2.72	0.85		
DP_4 variety	5.61			0.23		-3.05	2.73	0.19	-0.88
DP_5 quantity	0.35	-0.45	1.96	-0.39		3.49	0.62		-1.15
DP_4 quantity	4.18	0.25				-3.78	1.36		-0.68
Variable	Constant	requirement	use scenario	function	artifact	behavior	issue	isolated requirement	isolated use scenario

DP_5 max quality	0.18		-0.78	0.53	-0.22	-0.2	0.11		
DP_4 max quality	-0.06	0.1	0.6	-0.09	0.13		0.16	0.27	-0.91
DP_5 avg. quality	0.32		-0.73	0.68		-0.43	0.38		-0.71
DP_4 avg. quality	-0.04		0.41	0.05	-0.06	-0.03		0.2	-0.87
DP_5 max novelty	0.19			0.74	-0.85	-0.1	0.09	-0.46	
DP_4 max novelty			2.17	-0.11			-0.52	1.57	-2.53
DP_5 avg. novelty		0.17		0.36	-0.51	0.03			0.4
DP_4 avg. novelty	-0.12		1.38	-0.05	-0.2	-0.1	-0.41	0.97	-1.58
DP_5 variety	0.27	0.15	-2.13	-0.49	-1.8	0.81			0.4
DP_4 variety	-0.31		2	-0.11			-0.48	1.13	-2.42
DP_5 quantity	0.82	0.21	-2.75	0.35	-1.39	0.11	0.54		-0.38
DP_4 quantity	-0.23		2.86	-0.28		0.11	-0.16	0.66	-1.21
Variable	isolated function	isolated artifact	isolated behavior	disconnected requirement	disconnected use scenario	disconnected function	disconnected artifact	disconnected behavior	disconnected issue

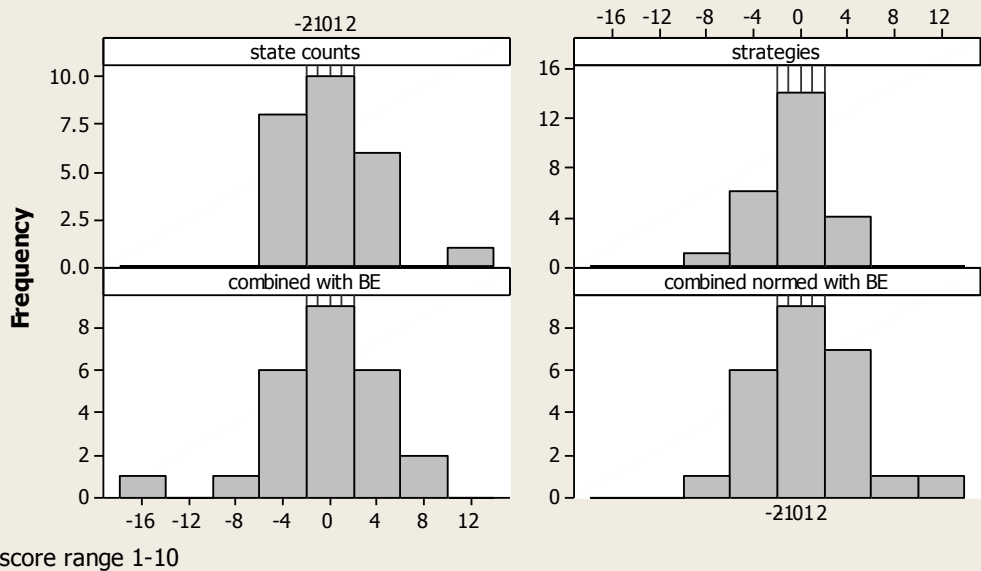
DP_5 max quality		0.58	-3.56	5.9	-10.8
DP_4 max quality	0.15		-0.92	0.52	-1.25
DP_5 avg. quality		1.13	-7.6	10.3	-21.5
DP_4 avg. quality		-0.1	-0.62	0.38	-0.9
DP_5 max novelty	-0.44	1.11	-2.57	1.96	-13.4
DP_4 max novelty			-4.3	2.36	-1.7
DP_5 avg. novelty	-0.29	0.65	-2.3	2.31	-3.36
DP_4 avg. novelty			-2.53	1.19	-2.5
DP_5 variety	-1.62	-0.36	11.5	-13.8	17.2
DP_4 variety	0.71				-4.03
DP_5 quantity	-1.39	1.28	1.44		-19.8
DP_4 quantity	0.83		0.43		-2.61
Variable	Abstraction	Entity depth prevalence	Order req_use	Order req_fun	Conflict identification

APPENDIX F

HISTOGRAMS OF PREDICTION RESIDUALS

Differences between actual and predicted quantity

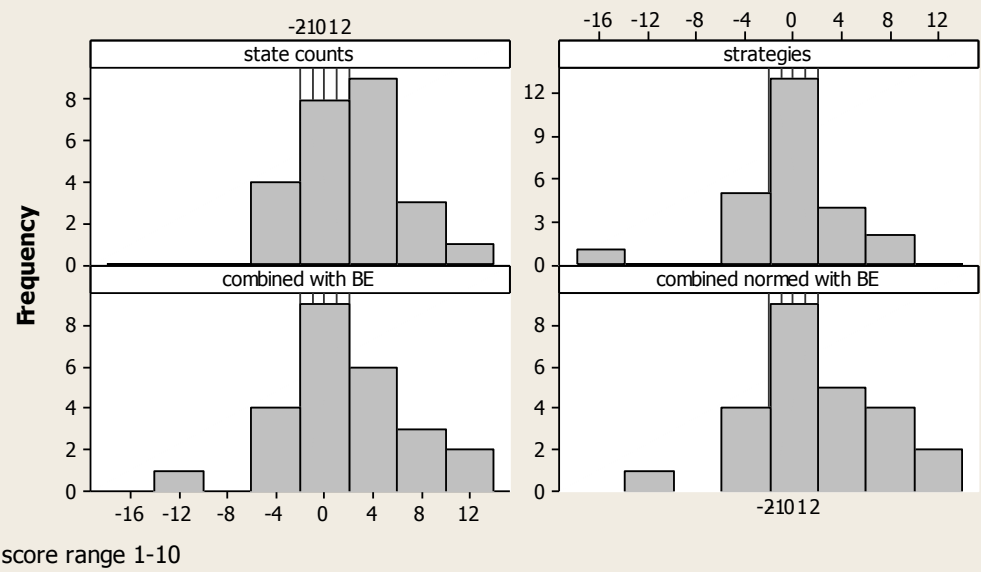
Model based on DP_4 predicting DP_5



score range 1-10

Differences between actual and predicted variety

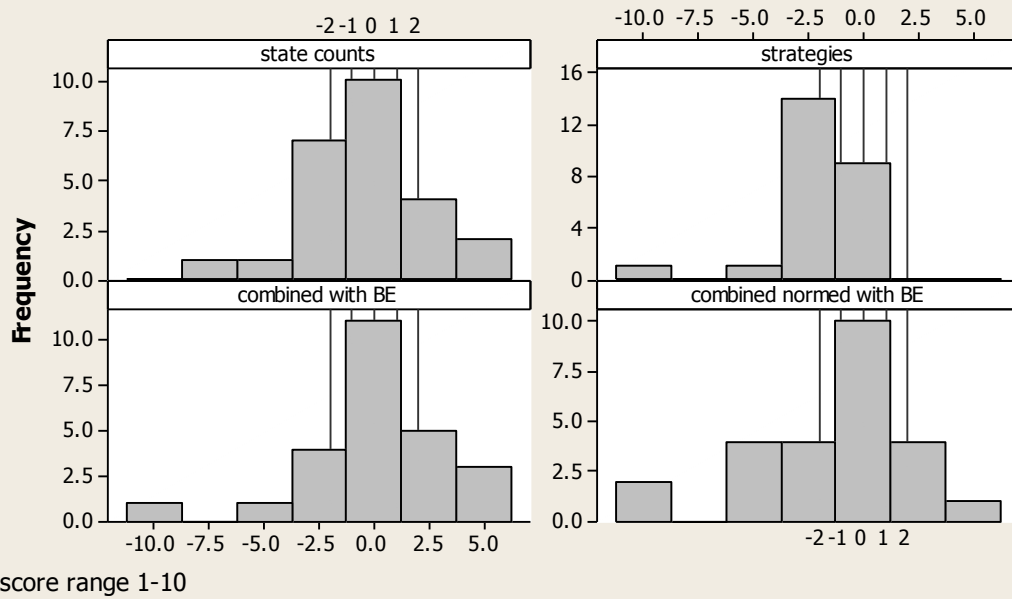
Model based on DP_4 predicting DP_5



score range 1-10

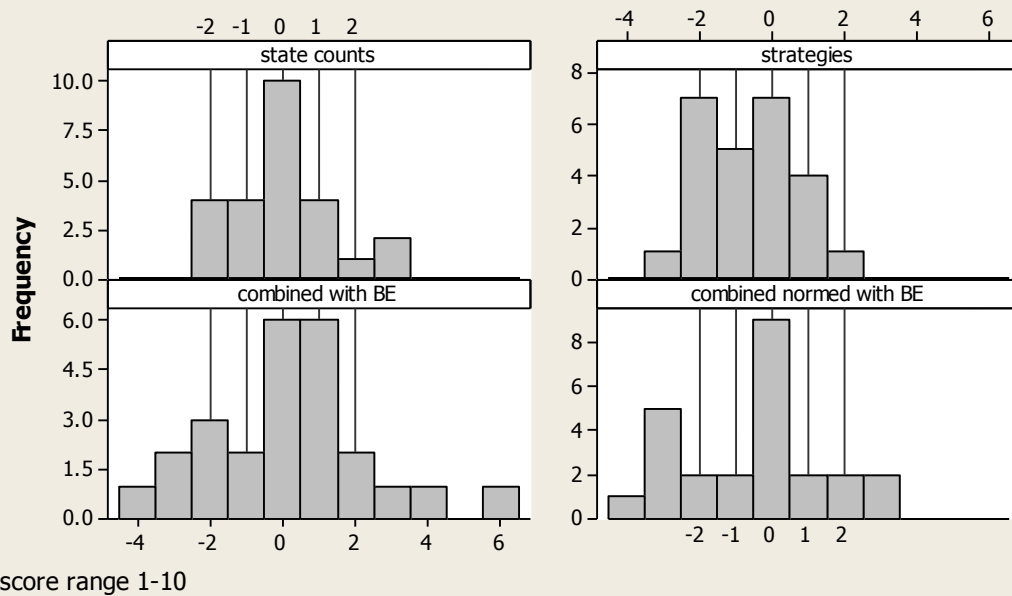
Differences between actual and predicted novelty

Model based on DP_4 predicting DP_5



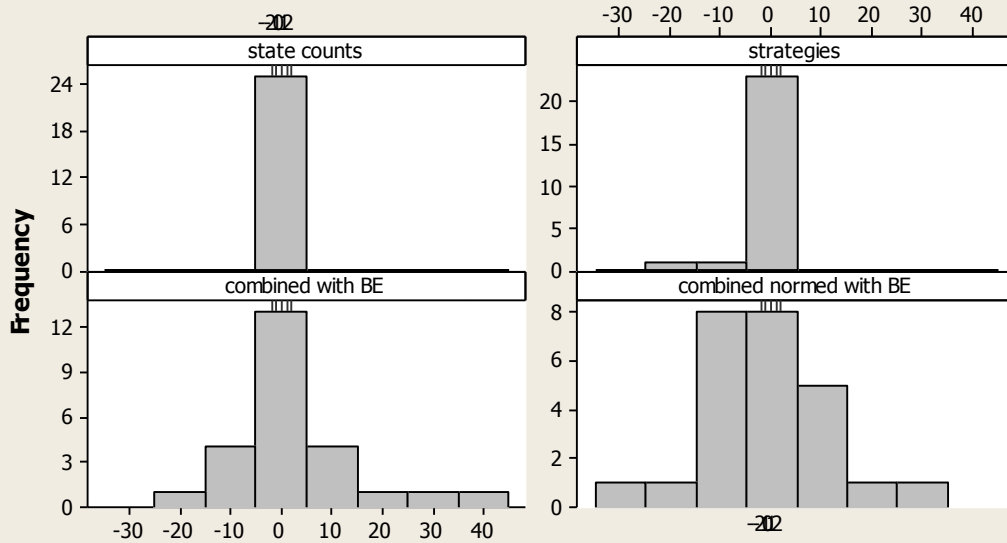
Differences between actual and predicted quality

Model based on DP_4 predicting DP_5



Differences between actual and predicted quantity

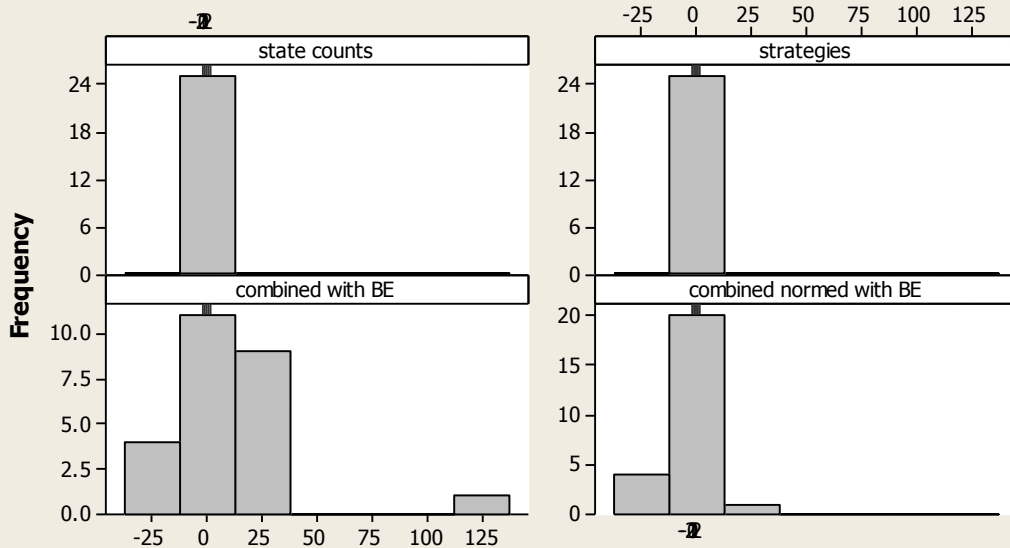
Model based on DP_5 predicting DP_4



score range 1-10

Differences between actual and predicted variety

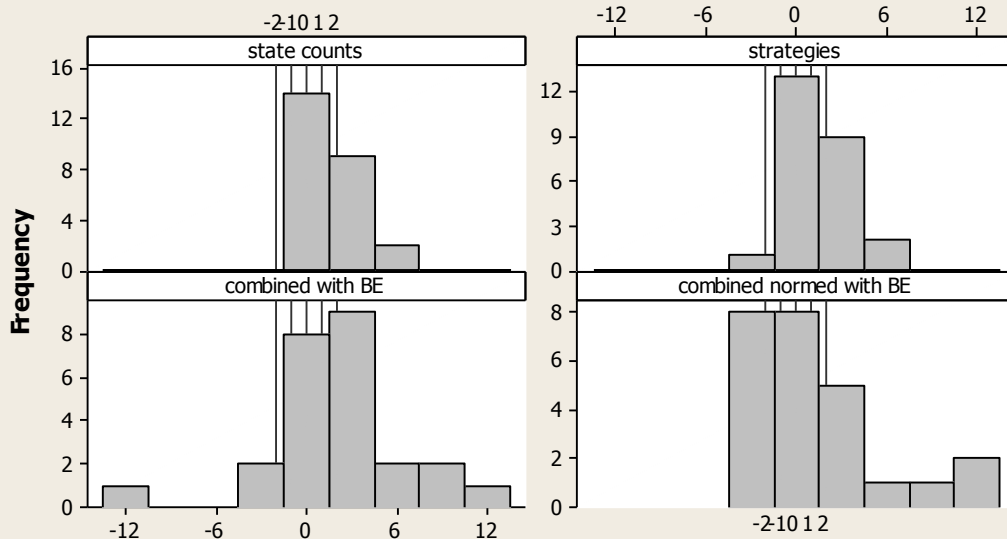
Model based on DP_5 predicting DP_4



score range 1-10

Differences between actual and predicted novelty

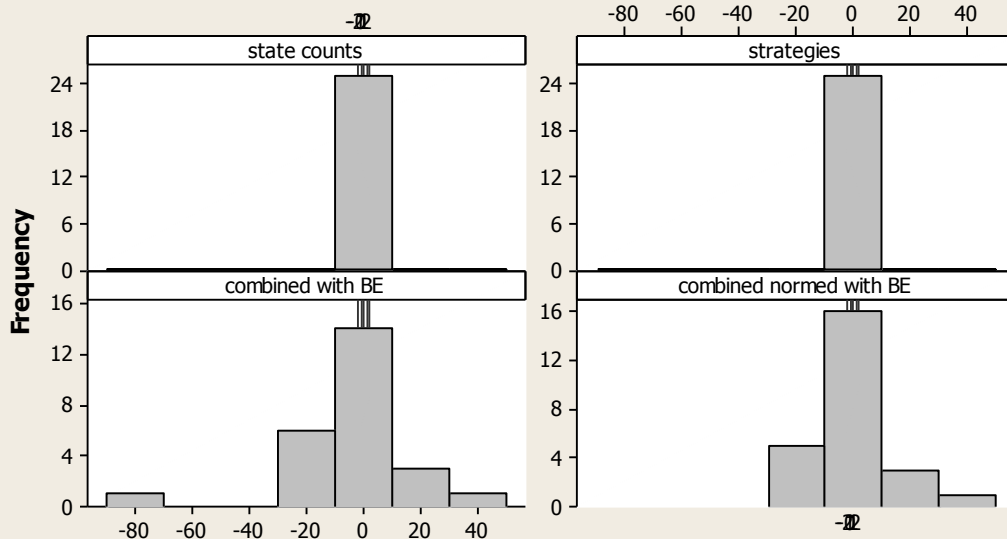
Model based on DP_5 predicting DP_4



score range 1-10

Differences between actual and predicted quality

Model based on DP_5 predicting DP_4



score range 1-10