A Methodology for Empirical Measurement of Iteration in Engineering Design Processes

Michael John Safoutin

A dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

University of Washington

2003

Program Authorized to Offer Degree: Mechanical Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Michael John Safoutin

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of Supervisory Committee:

_____

Cynthia J. Atman

Reading Committee:

_____

Cynthia J. Atman

_____

Vipin Kumar

_____

Robert P. Smith

Date: _____

In presenting this dissertation in partial fulfillment of the requirements for the Doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of the dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to ProQuest Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature _____

Date _____

University of Washington

Abstract

A Methodology for Empirical Measurement of Iteration
in Engineering Design Processes

Michael John Safoutin

Chair of the Supervisory Committee:

Professor Cynthia J. Atman
Industrial Engineering

Empirical study of the engineering design process typically involves the observation of design activity and the measurement and analysis of its features. An activity as complex as design possesses a wide variety of features, many of which are prominent and likely to attract research attention. The iterative feature of design activity is particularly prominent, but few measures are available to support its empirical study. In response to this need, measures of design iteration are developed that enable objective distinctions to be drawn among empirically observed design processes in terms of potentially meaningful aspects of their iterative character. A conceptual framework leads to the recognition of several varieties of design iteration and the selection of one variety of particular interest to design research. Several measures for this variety of iteration are developed and implemented through the application of metrics to timeline depictions of the design process. A data collection instrument is then developed and used to capture empirical data representing the design processes of two groups of subjects individually solving a design problem. The measures are then applied to the data, resulting in profiles of the iterative character of each observed process and the grouping of subjects into similar groups. The validity of the measures is supported by showing that the result of their application to each group is comparable and that their measurements are consistent with expectations regarding the nature of the design processes being observed.

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

Figure Number Page

LIST OF TABLES

Preface

This dissertation has a broad scope, beginning with the development of an original conceptual framework and proceeding with the derivation of new measures of iteration, development of a data collection instrument, collection of empirical data, and analysis of the data. The following "road map" may be helpful in navigating the document.

Chapter 1 describes the motivation for pursuing this topic. It points out the importance of empirical study of the design process and the need it creates for effective measures of various features of real design processes. The iterative feature of design is identified as a particularly interesting feature for which measurement methods are not yet sufficient to support its empirical study. The chapter concludes with an outline of the goals of the study, a description of similar studies, and description of a basis for establishing validity of the measures that are to be developed.

Chapter 2 begins by suggesting that empirical study of iteration requires a way to objectively extract it from empirical data, a task which involves its recognition and its measurement. A survey of existing literature is focused upon (a) the capability of existing conceptualizations of iteration to support its recognition, and (b) the capability of existing measures and representations of iteration to support its empirical measurement. It establishes the need for a precise conceptualization of iteration that is relevant to the motivations of empirical research and that can lead to effective measures.

Chapter 3 develops a conceptual framework that provides the groundwork for effective measures. The framework is based upon an understanding of design as a form of process. Properties of processes in general and of processes of design are contrasted and related to patterns of design activity that are commonly described as iterative. This leads to the recognition of several varieties of design iteration, which are then examined with respect to their relative degree of interest to experimental research. This judgement is based on arguments addressing three criteria: potential for variability, potential for influence on design outcome, and potential for a designer to control this influence in practice. *Feedback iteration* is judged to hold the greatest degree of interest, leading to the decision to focus measures on this variety of iteration.

Chapter 4 develops a measure of feedback iteration. It begins by adopting a model of design as an information processing activity operating with respect to a parametrically structured design problem. This model suggests several timeline schema for representation of observed design processes. Among the various event patterns that these schema can portray, several specific patterns are suggestive of a potential influence on outcome. These patterns are selected as the basis for a measure of iterative character that may be applied to timeline data.

Chapter 5 describes an approach by which a design task may be selected and instrumented in order to collect data sufficient to apply the measure. The approach is demonstrated in the context of converting an educational design software program called *Virtual Car* into a data collection instrument.

Chapter 6 describes how the instrumented software was deployed to collect data from two groups of subjects.

In Chapter 7, the measure is applied to this data. Evidence for validity of the measure is presented by showing that the result of its application to each group is comparable, and that their measurements are consistent with expectations regarding the nature of the design task that it measures.

Chapter 8 draws conclusions about the utility of the measure in empirical research, and outlines conditions for application of the measure to the study of other design tasks.

Acknowledgements

First, I would like to acknowledge the encouragement and support of those who influenced the last few years of my graduate work, which were the most educational and productive. My thesis advisor, Professor Cynthia J. Atman, acted as an ideal example of a faculty mentor. I learned a tremendous amount through my involvement with her research and through working with the people that she has brought together. When my original thesis advisor left the university, Cindy stepped in as a replacement, even though the busiest time of her career was about to begin. Her involvement made a unique impact on the end product and on my capability as a researcher. Professor John Kramlich of Mechanical Engineering not only introduced me to Cindy at a critical point in my graduate career, but also provided the opportunity to fulfill my longstanding desire to teach engineering design, as an instructor of ENGR 100. As a result, for the last three years my students have provided me with daily encouragement and have continually nourished my interest in design education and research. My original thesis advisor Rob Smith, being one of only a few department faculty having a specialty in design research, made it possible to consider pursuing the topic of this dissertation, and encouraged me to publish some of my findings early on. He agreed to continue as a committee member after his departure, and carried his involvement through to the end. Dr. Jennifer Turns provided me with perspectives on completing a dissertation, being a post-doc, and pursuing a faculty position. Mary Cook of the College of Engineering spearheaded the nominations that resulted in my teaching award. And the late Professor Dale Calkins was an advocate and friend, who inspired me early in my graduate career and made me feel that my work was valuable. I must also acknowledge my many graduate student friends from various departments of the university. Although many of them have already come and gone, their memory persists through their influence on various aspects of this dissertation. It was through our countless pub conversations that we found ourselves building an after-hours academic culture that was as close to bohemian as an engineering student may ever find. Their impact on the encouragement of my philosophical ideas and on my persistence in applying them to engineering design was profound, and I can only hope that in my future work I will continue to find and benefit from a similarly supportive environment.

1 Introduction

Engineering design has historically been known as a relatively unstructured art, compared to the scientific and technological disciplines that support it [Dixon 1987]. But as design projects in industry have become more complex and increasingly sensitive to economic and competitive pressures [Williams et al. 1995], questions about the nature of design activity and how it may be improved have grown in importance. The research specialty of design theory and methodology (DTM) has experienced rapid growth over the last twenty years in an effort to respond to these questions. By at least one account, books and articles on DTM have become the largest single category of publications on engineering design [Sullivan et al. 1994].

An increasingly large portion of DTM research is empirical [Gero and McNeill 1998]. It has long been observed that independent designers or design teams pursuing the same design problem may vary significantly in the processes they follow as well as the quality of their solutions [Braha and Maimon 1998]. This represents an opportunity to seek an understanding of what makes design processes effective by observing real design processes and comparing them in terms of their features and their end results. Design activity, however, is not naturally conducive to empirical study [Dixon 1987], [Valkenburg and Dorst 1998]. In the words of one researcher, "the study of the engineering design process is too complex for traditional study and analysis" [Stauffer et al. 1991]. One of the areas that has required significant attention is the development of effective methods for the observation and analysis of design processes [Oxman 1995].

1.1 Observation and Analysis of the Design Process

Empirical research is centered around observation of the real world and analysis of what is observed. For observing design activity, a variety of methods have been developed or adapted from those used in other fields [Stauffer et al. 1991]. These range from relatively informal methods, such as direct observation of design teams [Marples 1961], to retrospective methods such as interviews, questionnaires, or design diaries [Ball

et al. 1994] that take place after design activity, to real-time protocol methods which meticulously record verbal or other evidence of design activity as it takes place [Stauffer et al. 1987, 1991], [Ericsson and Simon 1993], [Dwarakanth et al. 1996].

Observation methods provide a means to collect evidence of design activity, but they must be supplemented by effective methods for analysis of the data they collect. An investigator typically wishes to focus analysis on specific features of the design process, perhaps those that represent dependent or independent variables relating to an experimental hypothesis, or those that may inform a specific research question. The features of interest must then be extracted from other activities that are likely to be represented in the raw data. The task of extraction calls for a way to reliably recognize the feature where it is represented in the data, and apply suitable measures to it. For example, if one seeks to relate a feature such as the "systematic nature" of a design process to another feature such as "degree of exploration of the solution space", it requires some way to recognize and measure both features in an empirical setting.

Some of the simpler features of design activity are relatively straightforward to recognize and measure, but others present a greater challenge. Features such as time spent designing, time spent in specific activities, and patterns of activities over time have been successfully extracted using verbal protocol methods [Chimka and Atman 1998]. But many other features that are equally familiar, interesting, and potentially informative lack a sufficiently precise conceptualization to allow their reliable extraction. For example, when attention was focused on design process features such as *design strategy* [Ball et al. 1994],[Fricke 1996] and *decision path* [Dwarakanth and Wallace 1995], these features first had to undergo more formal definition in order to facilitate their recognition, and new constructs had to be developed to support their measurement. The study of many other features of design activity encounters the same difficulty. As Love has pointed out, many of the core terms associated with design, including the term *design* itself, have developed so many implicit, imprecise, and overlapping definitions that "they potentially include so much that they no longer clearly define anything" [Love 2002, p.355]. This situation presents a challenge for progress in empirical research on design.

1.2 Need for Measures of Key Features of Design

As research activity in DTM generates an ongoing demand for methods to "get at" various features of the design process, new constructs and representations are continually being developed [Smith and Browne 1993, p. 1215]. Investigators who are concerned with a specific feature of interest may encounter the burden of developing and validating a means for its measurement before a specific hypothesis or research question regarding it may even be addressed. This prerequisite task, which typically is not trivial, consumes resources and may discourage research curiosity in areas that might be profitable to explore. Furthermore, the development of constructs on an *ad hoc* basis encourages a proliferation of alternative constructs whose generality and applicability outside of their original context may remain unclear. For example, the abovementioned constructs for design strategy and decision path were not explicitly validated by their authors for use beyond the studies for which they were developed. This and their relative obscurity suggests that others may be led to develop alternatives that they feel to be better adapted to their own specific purposes or perspectives. When different studies involving similar features of design do not share the same measurement constructs, it becomes difficult to compare them effectively and generalize from their respective findings.

The development of general, robust methods for the recognition and measurement of key features of design activity is thus an important goal unto itself. That is the goal of this research.

Design activity has many discernible features. It is difficult to judge *a priori* which features may eventually be demonstrated to carry the most valuable insights toward design outcome or other profitable issues. However, it can be argued that the best candidates for study are likely to share several attributes. First, features that are particularly familiar and prominent to those familiar with design activity are likely to attract a proportional degree of research curiosity. Second, the prominence of a feature suggests a potential significance to design outcome or other issues of practical interest. Finally, features that fit this description but have yet to achieve an established and effective means of recognition and measurement are obvious candidates. It may be

argued that any feature that is familiar, prominent, and difficult to extract from empirical data deserves attention toward the development of effective methods for its measurement.

1.3 The Iterative Feature of Design

One of the most familiar and prominent features of design activity is its *iterative* character. The iterative quality of design is widely recognized by designers (e.g. [Braha and Maimon 1997], [Urban and Hauser 1993, p. 173]), is acknowledged in a broad cross section of design research (e.g. [Madanshetty 1995], [Cohen et al. 1994], [Nukala et al. 1995]), and is portrayed prominently in nearly every model of the design process (e.g. [Finger and Dixon 1989], [Evbuomwan et al. 1996]) and textbook definition of design (e.g. [Beakley et al. 1986], [Dieter 1991], [Eide et al. 1998]). Articles on engineering design frequently allude to its iterative character in literally the first paragraph (e.g. [Kusiak and Larson 1995], [Dwarakanth and Wallace 1995]).

Motivations for studying iterative behavior are not hard to find. For example, in design organizations, a majority of development time and cost consists of what has been described as iterative activity [Osborne 1993], [Cooper 1993]. If this is true, then the iterative character of design represents a path for research toward a fuller understanding of design activity and ways to improve its outcome. Just as significantly, guidelines are needed to help manage iterative activity in practice. Wileden [1986] called for tools to guide iteration in software development processes, recognizing several needs including the need to make best use of information yielded through iterations, to determine when one should embark on an iteration, and to identify rework resulting from a proposed change. Curtis [1986] called for management tools to reduce the number of iterations in software design projects. Cooper [1993] calls for management methods to help plan, monitor, and reduce the magnitude and duration of rework, which is commonly associated with iteration. Eppinger et al. [1997] call for tools to "recognize and manage the iterative nature of the design process" leading to direct indications for managerial control, such as co-locating persons involved with highly iterative tasks.

Unfortunately, prevailing attitudes about the place of iteration in design suggest that these wishes are still far from being met. Little progress has been made toward achieving an understanding of iteration that is sufficient to provide a basis for answering these calls. In its place, one commonly encounters what could be described as a *zero-iteration ideal*, a default presumption that a non-iterative or minimally-iterative design process is always most desirable, if it could only be achieved. This presumption appears to be widespread and can be found in many places. For example, a passage in an NSF report on research opportunities in engineering design [NSF 1996, p. 5] states:

> "If the initial design is poor, several design-analysis iterations are necessary before a satisfactory design is found; the ideal goal is to eliminate iteration and have the design meet the specifications on the first pass",

and in his landmark text *The Principles of Design*, Suh [1990, p. 32] mentions:

> "The most desirable iteration cycle, next to no iteration, is the reiteration at the conceptual stage of the design process itself".

Admittedly, these remarks are rhetorical and are probably not intended as dictums, but they do suggest that the prevailing understanding of iteration focuses primarily on its negative aspects, and has yet to embrace a more balanced perspective that includes positive aspects as well. Empirical research is a natural path toward submitting the zero-iteration ideal to the rigorous examination it deserves, perhaps leading to a perspective that is more capable of providing concrete guidance for management of iterative processes.

A likely focus of empirical research on design iteration would concern relationships between the iterative character of a design process and its outcome. The various calls for management tools that were outlined above imply that some sort of relation between iteration and design outcome is commonly assumed to exist; for example, according to Browning [1998], viewing a development process from the perspective of its iterative character may help "capture and quantify drivers of cost, schedule, and performance variability".

In order to capture such relationships on which tools might be built, an investigator would first need to be able to capture the iterative character of observed design processes, and then apply measures that allow individual processes to be compared in terms of relevant aspects of their iterative character.

1.4 Can Design Iteration Be Measured?

Design iteration has yet to form a consistent, precise and widely accepted conceptualization that would support the development of objective and reliable measures. It lacks an established theoretical or operational definition, and its lay definitions are imprecise and inconsistent. Iteration has been associated, equated, or used interchangeably with a diverse set of implicitly related terms such as repetition [Curtis 1986, Eppinger et al. 1997], refinement [Urban and Hauser 1993, p. 171], rework [Cooper 1993], redesign [Eisenhardt and Tabrizi 1995, Terwiesch and Loch 1999], backtracking [Tully 1986, Ullman et al. 1988], recursion [Ward 1990], patching [Ullman et al. 1988], and the need to "try again" [Thomke et al. 1998, Thomke 1998]. These terms and references make sense in their original contexts, but taken together they delineate a conceptual space that is too broad to readily suggest operational strategies for effectively recognizing and measuring iteration in an empirical setting.

Under these conditions, research that has attempted to relate iteration to outcome has not achieved the degree of conclusiveness that would suggest clear advice for designers. For example, with regard to design time, Smith et al. [1992] found that students who followed a "standard" strategy recommended in the instructions to a design-based game (a strategy described as relatively iterative) took more time to find a solution of equal quality than those who used a reordered task strategy meant to minimize the potential for iteration. Another study by Terwiesch and Loch [1999] also related the presence of fewer iterations to reduced design time. However, Eisenhardt and Tabrizi [1995] had drawn the opposite conclusion in showing that development processes having *more* iterations tended to require less time, even though both studies had employed the same definition of iteration.

Studies such as these are attempting to relate an independent variable called "iteration" to a dependent variable such as "design time" or "design quality". But does the independent variable "iteration" measure the same phenomenon in each study? Although the studies of Terwiesch and Loch [1999] and their colleagues Eisenhardt and Tabrizi [1995] share the same conceptualization of iteration, one explanation for their contradictory results might be that this conceptualization includes somewhat different types of specific iterative behaviors that have different effects on design time. Similarly, the concurrence of Smith et al. [1992] with Terwiesch and Loch [1999] should be taken with a caveat because the two studies define and capture iteration in different ways (the first by means of a protocol analysis of revision operations in a design game, and the second by a survey of industry designers who were provided with a conceptual definition of iteration). While either basis is reasonable, it remains unclear whether they capture the same sort of "iteration". Studies that are affected by this issue are unlikely to contribute as effectively as they might toward building a cohesive literature on iteration because they lack comparability. One simply cannot be certain that they all measure the same thing.

The prospect of "getting at" the iterative character of observed design processes in a way that supports comparability of findings continues to pose a methodological hurdle to empirical researchers. Existing conceptualizations do not support the ability to draw objective distinctions among design processes in terms of relevant aspects of their iterative character. An objective measure of design iteration that may be employed for this purpose is an imperative for effective empirical investigation of its significance to design activity.

1.5 Goals and Methodology

This study is concerned with the question,

How may one objectively draw distinctions among observed design processes in terms of relevant aspects of their iterative character?

This study seeks to develop and validate measures of iteration that are suitable for use in empirical research. Any experimental study that would seek to relate iteration to outcome would first call for an effective means to measure iteration as an independent variable. This study is therefore concerned with developing measures that may be applied to this purpose.

The intent of this study is comparable to that of other studies that have sought to characterize a specific feature of the design process, develop means for its measurement, and provide evidence for validity of the measure. For example, Goel [1994] developed a conceptual framework concerning design and non-design problems and hypothesized a set of invariant features specific to design problems. Representations of empirically captured design and non-design processes were then developed, and used to support the contentions of the conceptual framework that design and non-design processes are different in the ways asserted. Validity of the framework and representation method was supported by demonstrating that they led to useful and explanatory results. In another study, Gunther and Ehrlenspiel [1999] sought to measure the systematic nature of an observed design process. They captured and represented the design processes of three design teams who were variously instructed with regard to adherence to a systematic model of the design process. Their representations successfully distinguished the relatively systematic process of the group that was told to design systematically from the less systematic processes of the other groups. Validity of the representation method was supported by demonstrating agreement between the differences that were expressed by the representations and the differences that were expected to result from differences in instruction. Another example is found in Austin et al. [2001], where a conceptual framework was developed to suggest a classification of activities involved in conceptual design tasks. Representation methods based on the activity categories of the framework were then employed to map the conceptual design activity of a group of subject design teams. The framework found initial validation in finding that the great majority of activity observed in the experimental task was successfully classified within the categories of the conceptual framework.

The design of the current study is depicted schematically in Figure 1.1. It consists of three major stages: *development* of measures of iteration, *application* of the measures to empirical data, and *validation* of the measures.

The *development* phase consists of the following components. Existing conceptualizations of the iterative feature of design activity are reviewed and compared, leading to the development of a conceptual framework that formally defines several varieties of iteration and selects one variety that is judged to have the most relevance to the goal of relating iteration to outcome. Measures are then developed that may be applied to empirical data. A data collection instrument is then developed with which to capture the design processes of subjects conducting an individual design task.

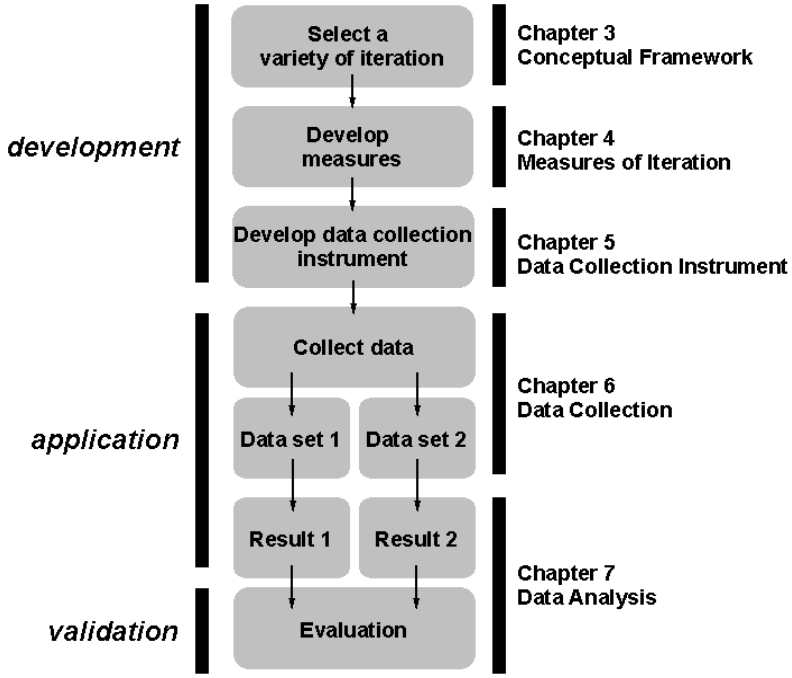


Figure 1.1. Study plan

The *application* phase consists of using the instrument to gather two data sets from two groups of subjects, and application of a selected measure to the data. This phase results in two sets of design processes that have been profiled by application of the measure, allowing comparisons to take place within each group and between the two groups.

The measure is then validated by evaluating its performance with respect to the two data sets. One perspective on the issue of validating a measure suggests that validity is not proven *per se* but is supported by a validity argument, which consists of supporting evidence and explanatory rationale that accumulates over time [Cronbach 1988]. In this study, a foundation for a validity argument is built upon evidence that the measure has replicably provided measurements that are consistent with what would be expected given the nature of the assigned task and the subjects. This leads to the conclusion that the measure was effective at measuring real differences in the iterative character of the processes to which it was applied, and provides research utility in its ability to identify and quantify such differences in an objective manner.

2 Literature Review

Calls for management tools to assist design activity might be answered by identifying relationships between influential features of a design process and its outcome. Such relationships are presumed to operate behind many well accepted design formulations. For example, prescriptive models of design are an expression of a presumed relationship between some aspect of the process by which design is pursued, and some aspect of outcome, such as time to completion [Duffy and Salvendy 1999, Calantone and di Benedetto 2000], design cost in terms of money or time [Pahl and Beitz 1988], [Delaney 1997], or the quality of the product delivered [Taguchi 1986], [Suh 1990].

The iterative aspect of process is simply one aspect through which the influence of process on outcome might operate. If it does, then those seeking to improve their design process would do well to learn how to manage iteration to best effect. Although prescriptive models of design provide a variety of guidance for the planning and execution of the design process, they provide little specific guidance for managing its iterative character. A research plan toward this goal might consist of the following:

- Empirical observation of real design processes,
- Measurement of relevant aspects of their iterative character,
- Drawing of distinctions among the processes based on these measures,
- Relation of these distinctions to the respective outcomes of each process.

An ideal environment for this effort would fulfill the following "wish list":

- A reliable and objective measure of iterative character is available,

- A plausible argument exists to suggest that the sort of distinctions in iterative character that the measure describes relate to design outcome, and

- The measure is applicable to a variety of design problems, so that findings may be replicated in different settings.

Effective measures are key to empirical study of iteration. A key component of its measurement is its recognition, in order that it may be captured and subjected to measurement. The remainder of this chapter reviews current knowledge related to the *recognition* and *measurement* of the iterative character of observed design processes.

2.1 Can Iteration Currently Be Recognized?

In order to recognize any feature of design activity, one must begin with a clear concept of the feature itself. A broad variety of conceptualizations of iteration have been expressed in the design literature.

2.1.1 Concepts and Definitions of Iteration

One first encounters descriptions of iteration in textbook models and definitions of the design process, nearly all of which feature prominent iterative loops that link several distinct stages [e.g. Pahl and Beitz 1988, Cross 1989]. Sometimes these loops are likened to a feedback control mechanism [e.g. Taguchi 1986, Suh 1990]. Iteration is typically described as an inevitable correction and revision process and illustrated by examples of design errors and delays.

A sample of articles that touch upon the iterative aspect of design reveals a number of more specific perspectives. Some focus on perceived attributes of iterative activity, such as its obvious repetitive aspect [Curtis 1986, Eppinger et al. 1997] in which one is frequently seen to "try again" [Thomke et al. 1998]. Others focus on the apparent function of iterative activity, associating it with the refinement of a design [Urban and Hauser 1993, p. 171]. Still others find it useful to focus on specific manifestations of iterative activity such as *rework* [Cooper 1993] or *redesign* [Eisenhardt and Tabrizi 1995, Terwiesch and Loch 1999]. Others have associated iteration with specific behaviors such as *backtracking* [Tully 1986], *patching* [Ullman et al. 1988], *recursion* [Ward 1990], and *transitioning* among classes of design activities [Atman et al. 1999], [Adams et al. 2001]. Several distinct varieties of iteration have been recognized, based on criteria such as whether the iteration was expected or unexpected, or whether it occurred in a parallel or

serial manner [Smith et al. 1992], [Smith and Eppinger 1993, 1997]. Similar distinctions have formed the basis of at least one classification of iteration varieties [Safoutin and Smith 1998].

Conceptual definitions of iteration can often be found in such work. Gebala and Eppinger [1991] define iteration as "the revision of decisions which had been made using incomplete or imperfect information". Nukala et al. [1995] and Eppinger et al. [1997] define it as "the repetition of activities to improve an evolving design". Ford and Sterman [1998] define it as "work on tasks to make changes subsequent to their initial completion". Adams [2001] defines it as "a goal-directed process that utilizes reasoning processes and strategies to gather and filter information about the problem, monitor progress, and inform the generation or revision of possible solutions". While these definitions seem generally agreeable, they do not readily suggest efficient ways to identify events that meet these definitions.

2.1.2 Need for Operational Definitions

The problem of capturing a variable that is to be measured may be understood as one of finding an effective *operational definition*. In contrast to a *theoretical* (or conceptual) definition, which defines a variable in relatively abstract terms, an operational definition is one that defines a variable "in terms of the operations necessary to measure it in any concrete situation" [Rosenthal and Rosnow 1984].

Existing definitions of iteration such as those reviewed above are primarily theoretical and as such do not provide sufficient conditions for the application of measurement. They tend to be either too specific to apply universally or too general to apply efficiently. For example, Eisenhardt and Tabrizi [1995] gather data about iteration via a questionnaire in which iteration is defined as the redesign of ten percent of a product's parts. Like similar definitions, it is effective in its specific context but is too specific to apply to many others, such as those in which design of discrete parts is not involved. A specific definition is also likely to be too narrow to apply universally. In a closely related study, Terwiesch and Loch [1999] use the same definition and point out that it excludes

activities such as "debugging", which most would describe as an iterative activity. On the other hand, a more general definition may exclude less, yet isolate less, meaning that recognition is likely to be more laborious. For example, Adams [2001] used a conceptual definition of iteration to develop coding rules for interpreting design stage transitions in verbal protocol data as iterations. Although the rules and the definition on which they were based provided a way to recognize iterations quite effectively, application of the rules still required a laborious process of manual coding.

In the field of software design, Tully [1986] provides a relatively precise definition of iteration that begins to approach our criteria for operational measurement but unfortunately has shortcomings when applied to an engineering design context. Defining iteration, the definition refers to anything that decomposes into "repeated operations acting on and/or producing repetitive information" and has a "clearly definable termination condition". Several examples are provided: "entering named objects into a data dictionary", "compiling a set of program modules", and "reaching agreement in discussion of requirements with end users". Examination of these three examples reveals that each is quite different in its potential influence on outcome. Entering items into a dictionary is simply repetition of the same operation, but compilation of program modules involves successive refinement of a first pass and second (optimizing) pass, which is indeed repetitive but involves a set of fundamentally different operations, as well as a more complex termination condition and a refining effect on the product. Discussing requirements with end users is likely to be dominated by feedback, leading to the potential for revision of original goals or undoing of prior progress, which are not likely to be present in either of the other two examples. Thus, even this relatively precise definition is too inclusive to distinguish among these potentially significant behaviors.

2.2 Can Iteration Currently Be Measured?

Most previous efforts to quantify iteration in observed processes have employed various forms of design process representation. Research on a social activity such as design frequently leads to the seeking of patterns in depictions of qualitative data [Chi 1997]; and depicting design process data often involves the use of a design process

representation. For example, Olson et al. [1996] sought patterns in data captured from team design meetings by a combination of heuristic and statistical techniques applied to representations of observed design processes.

In the simplest terms, a representation is simply a depiction of something [Smith and Browne 1993], or "something that stands for something else" [Palmer 1978]. As a key tool for the solution of problems [Simon 1969], representations are applicable to engineering design for representing the design artifact [Bodker 1998], [Sobek 2001] as well as the design process [Banares-Alcantara 1995], [Chimka and Atman 1998]. A representation of a phenomenon may be constructed upon a subset of its structurally significant features and still be effective [Smith and Browne 1993]; in fact, many are. For example, written language is effective as a representation of thought although it does not convey its every feature.

Representations of the design process appear in several forms. *Models* of the design process are perhaps the most familiar. Models vary in their level of abstraction; *general* models express the overall structure of a process to contrast it with other types of processes, while *specific* models express instances of a process type to contrast them with other instances of the same type [Tully 1986]. Descriptive and prescriptive models of design [Finger and Dixon 1989] and project management charts such as Gantt and PERT charts [Stilian 1962] are examples of general and specific model representations, respectively. In contrast to model representations which are primarily forward-looking abstractions, representations are also employed to express a *history* of an actual instance of a design process as it was implemented. Design histories [Shah et al. 1996] and design timelines [Chimka and Atman 1998] are examples of process history representations.

2.2.1 Matrix representations

One type of model representation frequently encountered in studies relating to iteration is the *matrix representation* [Kusiak et al. 1995], [Blandford and Hope 1985]. A design problem is divided into subtasks which are listed in a preferred sequential order on the horizontal and vertical margins of a grid. A mark in the grid indicates that the task to

the left of the mark is dependent on the task above the mark. For example, referring to Figure 2.1, the mark at the intersection of row b and column c indicates that task b is dependent on the outcome of task c. Generally, marks that exist above the main diagonal indicate that an upstream (earlier) task is dependent on the result of a downstream (later) task. Hence, matrix representations highlight task interdependency, a situation that suggests a potential need for rework and hence iteration among subtasks. Knowledge of the potential for rework may be used to reorder tasks to minimize this potential, presumably minimizing the potential for iteration.

	a	b	c	d	e	f	g
a	•						
b	X	•	X				
c		X	•				
d			X	•			
e				X	•	X	
f				X	X	•	
g					X	X	•

Figure 2.1. Example matrix representation

2.2.2 Can Matrix Representations Measure Iteration?

Matrix representations have been explored at length under the name Design Structure Matrix (DSM) [Steward 1981, Gebala and Eppinger 1991] and have been used to model dynamic aspects of design projects, particularly relating to rework or iteration. In most of these applications, the matrix method has been used to *model* the *potential* for iteration in a hypothetical design process plan, rather than to represent observed iteration. However, matrix methods have occasionally been associated with representation of iteration observed in an actual design process. Fricke [1996] created a transition matrix showing the number of direct transitions that took place between each possible pair of a set of tasks in an observed process. The matrix made it possible to see the existence and frequency of "forward" and "backward" jumps between tasks, relative to an assumed ideal task sequence. Patterns of forward and backward jumping were used to indicate the

degree of deviation between the actual task visitation sequence and the ideal sequence. Austin et al. [2001] also represented observed processes via transition matrices. Transitions in which tasks designated as "earlier" (again, in an assumed ideal task sequence) were revisited after "later" tasks were then isolated and interpreted as evidence of iterations caused by interdependence of tasks.

Matrix representation of observed iteration becomes questionable when one realizes that reasons for backward transitions can vary. For example, backward transitions could be generated if the designer simply chose to complete portions of a task incrementally over several visits, perhaps in reaction to a constraint such as resource inavailability, rather than the need to address another task on which it is dependent. These are two different circumstances which might not both fit a given definition of iteration.

The ability to even construct a matrix representation of an observed process is highly dependent on the degree to which the design problem may be broken down into discrete constituent subtasks and a meaningful sequential order imposed on them. In many cases a task breakdown may be difficult to reconstruct from the thousands of often cryptic events that could compose an observed process. A matrix representation also imposes the influence of an ideal or recommended ordering for the tasks, an ordering which is likely to be somewhat arbitrary because, as Simon [1969] has pointed out, design problems have no predetermined solution path. Furthermore, jumps backward and forward may only be expressed in an aggregate manner in a matrix representation; individual jumps and their patterns over time cannot easily be expressed.

2.2.3 Timelines

Another way to represent an observed design process involves placing its constituent events into categories and plotting them across a time domain. This creates a two-dimensional representation with time or sequence on one axis and categorized events on the other. This is commonly referred to as a *timeline* [Chimka and Atman 1998]. As an example, if the data source is a verbal protocol, the events are verbal statements that have been systematically categorized according to a coding scheme. A generic timeline representation is illustrated in Figure 2.2.

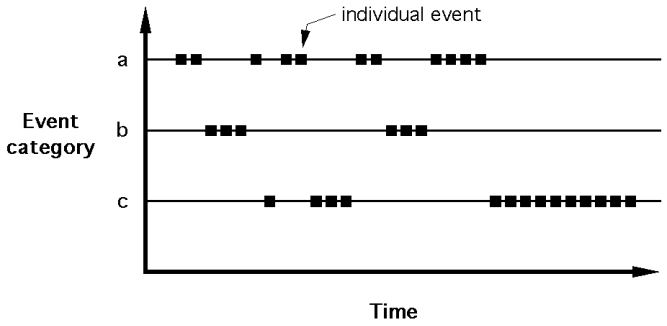


Figure 2.2. Example timeline representation

Timeline representations are a popular route for seeking patterns in captured processes, and can express certain types of distinctions very well. For example, Atman and Bursic [1998] represented instances of design problem solving in terms of sequential transitions among classes of design activities, resulting in representations similar to that depicted in Figure 2.3. They then related differences visible in the representations to the level of educational development of the designer [Atman and Bursic 1998, Atman *et al.* 1999]. Similarly, Gunther and Ehrlenspiel [1999] represented observed design processes in terms of visitations to four major design phases over time, in a format similar to that of Figure 2.4. These representations were then used to determine whether subjects who were formally trained in systematic design methodology followed a more systematic process than those who had developed their design skills solely via professional practice. For these authors, *design activity* and *design phase* were effective event categorizations, because they led to representations that highlighted features and patterns that were informative to their research questions.

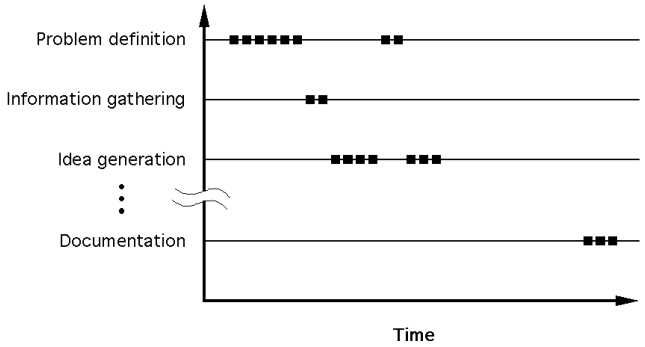


Figure 2.3. Events categorized by activity class

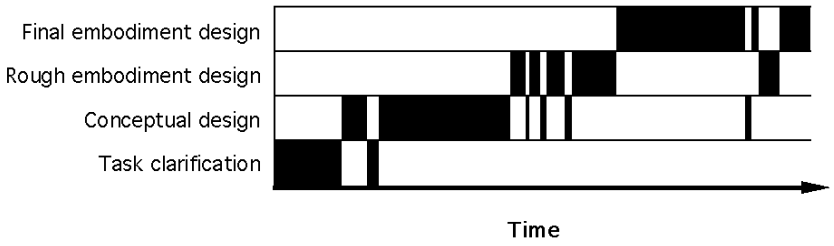


Figure 2.4. Events categorized by design phase

Event categorizations vary widely. Events have been categorized by the category of design activity they belong to [Atman and Bursic 1998], the level of abstraction of the artifact at the time of the event [Goel 1994], and by their function-behavior-structure (F-B-S) classification [Gero and McNeill 1998]. Some have categorized verbal events as being process-related or content-related [Stempfle and Badke-Schaub 2002]. Valkenburg and Dorst [1998] coded events according to the four categories of design actions espoused by Schon [1984] in his theory of reflective practice: *naming* (identifying a subfunction to consider), *framing* (beginning to consider the subfunction), *moving* (taking experimental actions to advance the design), and *reflecting* (reflection on the earlier activities to determine the next step). Christiaans and Dorst [1992] used the categories "Reflecting", "Sketching", "Gathering Information", and "Reading the Brief".

The choice of an event categorization depends on the research question that is being asked, and the feature or pattern that is most informative toward answering it. Differences in the basis of categorization changes the unit of the dependent axis, causing different types of patterns and distinctions to become visible.

Sometimes the primary purpose is to draw comparisons between individual processes. Malhotra et al. [1980] recorded client-designer dialogs and categorized the topics of spoken statements according to level of design abstraction (goal statement and elaboration, solution outline, elaboration, and explication, and solution agreement). By representing these dialogs along a time domain, they were able to discern a distinct cyclical pattern. Gero and McNeill [1998] employ a number of different categorizations, including level of abstraction (system level, subsystem interaction level, and two levels of

detail regarding specific subsystems), Function-Behavior-Structure classification, and what are called micro- and macro- strategies (characterizations of strategy at two levels of abstraction). They were able to show that substantially different problems resulted in substantially different representation profiles.

Patterns in represented processes may also be employed for normative comparison. Studies of this type seek to compare actual processes to idealized or "control" processes, for instance, those that are felt to implement the recommendations of a prescriptive model of design. Ball et al. [1994] kept track of activity transitions and whether they were consistent with or deviated from a normative prescription, and thereby was able to create representations that indicated how systematically the designer behaved in following a suggested design approach. In a later study Ball et al. [1997] represented observed processes in terms of the mean percentage of time spent discussing various design issues over several time periods, with the purpose of seeing how closely the designers followed normative standards suggested by a systematic design method.

One may also draw comparisons to see if discernible differences can be related to other things, such as characteristics of the designer that generated them. For example, Atman et al. [1999] related differences in represented processes to the subject's level (freshman or senior) in an engineering program. In another study, similar differences were used to measure the degree to which the subject's process had been impacted by the reading of a passage from an engineering text [Atman and Bursic 1996].

2.2.4 Can Timelines Measure Iteration?

In timelines that employ an event categorization based on a more-or-less sequential model (such as a traditional design phase model), one may discern patterns that are tempting to associate with iteration. For example, in representations employed by Gunther and Ehrlenspiel [1999] one may clearly see that designers often revisit "earlier" phases of design such as conceptual design, after having spent time in "later" phases such as embodiment design. In the representations of Atman and Bursic [1998], one can easily see patterns in which design activities commonly associated with early stages of design

are revisited repeatedly in later stages of the process. Fricke [1996] used timelines to show that the process of designers pursuing what was called a function-oriented strategy was characterized by a large number of "backward jumps" from concrete to abstract design phases. Are phase visitation patterns such as these depicting iteration?

Phase visitation patterns are difficult to confidently identify as iteration for some of the same reasons that cast doubt on matrix methods. Suppose that one has defined iteration as "the revisitation of previously completed work for the purpose of correction". Redesign of a misdesigned component would then qualify as iteration. Looking at the backward jumps in the timelines of the Fricke [1996] study, these jumps from the concrete to the abstract could be generated either (a) by redesign of a component, or (b) by a top-down, depth-first approach in which the designer fully designs one component and then fully designs another. No redesign takes place in the latter case, although backward jumps from the concrete (conclusion of component 1) to the abstract (start of component 2) would be generated. Similar backward jumps would be generated if the designer were to resume the design of a component that had previously been begun and then postponed.

Despite this interpretive ambiguity, phase visitation patterns in timelines are commonly taken as representing iteration (e.g. [Austin et al. 2001]). Under some potential working definitions of iteration, this assumption would be difficult to accept -- for example, if one were to interpret the visitation patterns reported by Austin et al. [2001] or Gunther and Ehrlenspiel [1999] as iteration under the example definition above, one must assume that every transition from a "later" stage to an "earlier" stage was for the purpose of correction, although this information is not available to support the assumption. Naturally these concerns might be resolved by manually interpreting each event to determine the actual context of the activity, but again the need for this additional step adds to the cost of data collection and analysis.

2.3 Summary: Need for Operational, Objective, and Relevant Measures

Neither matrix methods nor timelines have thus far provided a sufficiently effective means to measure observed iteration in an empirical setting. Measures that have been employed in previous work in this area can be susceptible to interpretive ambiguities and can be very costly to implement. Empirical study of iteration continues to call for a definition of iteration that is operational toward its measurement in an empirical research setting, that leads to reliable and objective measurement, and yields measures that are potentially meaningful to issues of research interest such as design outcome.

3 Conceptual Framework

The goal of this chapter is to identify an aspect of design iteration that is potentially meaningful to the interests of design research, and develop for it a precise conceptualization that can lead to objective measures. Toward this end a conceptual framework is developed to address the following goals:

(1) To acknowledge and define design behaviors that are commonly taken as iterative;

(2) To evaluate the potential interest of these iterative behaviors to design research; and

(3) To select the most potentially interesting variety for the development of measures.

The conceptual framework begins by examining design as a form of process. Properties of processes in general and of processes of design are contrasted and related to patterns of design activity that are commonly described as iterative. This leads to the recognition of several varieties of iteration, which are then examined with respect to their potential degree of interest to experimental design research.

3.1 Nature of Processes

3.1.1 Goal, Actions, and Outcome

A *process* is defined as a series of actions or events conducing to a goal [Thro 1991], [Merriam-Webster 1987]. Two properties of a process are therefore its set of *constituent actions* and its *goal*. A goal represents a desired state of affairs that the constituent actions of the process are meant to bring about. Although a process orients toward a goal, its conclusion does not necessarily indicate achievement of the goal. The conclusion of a process is best understood as delivering an *outcome,* which represents an actual conclusion relative to whatever the goal has become [Gotlieb 1992]. An outcome

is therefore one specific instance of reality that, it is hoped, falls within the desired state of affairs as delineated by the goal statement at conclusion.

3.1.2 Processes of Design

If design is a form of process, then *design* processes also possess a goal, a set of actions, and an outcome. The goal is commonly expressed in the form of a set of desired functional requirements to be delivered by the artifact [Suh 1990]. Outcome is commonly understood in terms of cost, time, and quality [Delaney 1997]. Specifically, design cost and design time represent an expenditure of resources in trade for the desired function, while design quality represents the functionality achieved relative to the desired functionality.

The constituent actions of a design process are more difficult to characterize. At a very rough level, descriptive process models of design characterize the constituent actions of design in terms of a series of roughly sequential design stages. But this perspective offers little insight into constituent actions within individual stages, where activity becomes more complex. A typical design process consists of a diverse array of actions of widely varying significance, many of them taking place concurrently and often leaving little physical evidence of their occurrence. This makes the characterization of design actions at anything more than a nominal level of granularity a daunting task. But the perception of design as an *iterative* process means that at least some of its constituent actions take on a pattern that is perceived as iterative. Exactly what does one perceive in judging the constituent actions of a design process as being iterative? An examination of existing concepts of iteration may illuminate this question.

3.2 Repetition Iteration

As an example of a process, consider the goal of sending a message to someone. One process for achieving this goal might consist of the following constituent actions:

(1) get a writing instrument,
(2) get a sheet of paper,
(3) write the letter,
(4) fold the letter,
(5) get an envelope,
(6) address the envelope,
(7) place the folded letter in the envelope,
(8) seal the envelope,
(9) drop the envelope into a mailbox.

These nine actions could of course be decomposed into subactions; for example, "seal the envelope" could be decomposed into "unfold flap", "moisten gum", "refold flap", and "apply pressure". For the purpose of illustration, these nine actions describe the process quite adequately.

This process, like many processes depicted at this relatively broad level of detail, has an orderly, straightforward quality. It proceeds incrementally toward its conclusion as each action is successively completed. Although each action may vary in its importance to the goal or in the degree of effort it requires, the successive traversal of actions conveys a sense of incremental progress toward the conclusion. To refer to this incremental property in which constituent actions successively increment a process toward a conclusion, the term *process incrementation* will be employed.

In a process that is as orderly and straightforward as the letter-sending example, it is difficult to detect any quality that might be called iterative; iterative processes seem to possess something more.

Most concepts of iteration acknowledge or imply a repetitive character. The dictionary defines *iterative* simply as "involving repetition" [Merriam-Webster 1987]. Nukala et al. [1995] and Eppinger et al. [1997] define iteration as "the repetition of activities to improve an evolving design". Tully [1986] emphasizes "repeated operations". A host of other concepts and definitions associate iteration with implicitly repetitive activity such as *re*vision, *re*work, *re*design, and "trying *again*". The perception of repetition in a process, then, is enough to assign it a basic sort of iterative property. Returning to the dictionary, one finds that *repetition* (i.e. to *repeat*) is "to make, do, or perform again". This definition suggests that a minimum requirement for repetition is the

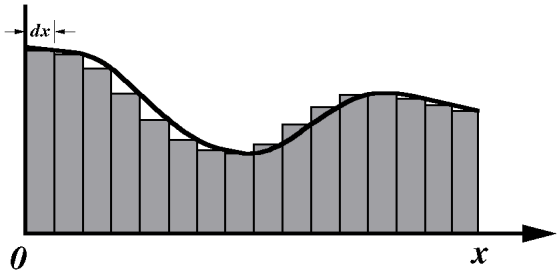
existence of multiple instances of an event (i.e. *made, done, or performed*) that take place in succession (i.e. *again*). Thus, repetition might be said to exist wherever there is a relationship among distinct actions in terms of similarity and temporal succession.

Even if a process does not appear to have a repetitive component at one level of detail, repetitive patterns might become visible when actions are decomposed further. For example, consider this process for making coffee in an automatic coffeemaker:

(1) grind coffee,
(2) install paper filter,
(3) scoop three ounces of coffee into filter basket,
(4) add water to fill line,
(5) activate power.

Although this process is not at first visibly repetitive, the act of placing coffee in the filter would decompose into three repetitive scooping actions if the scoop has only a one-ounce capacity. Likewise, in the letter-sending example, the act of sealing the envelope may involve several successive licks of the gum, and several successive applications of pressure to the seal. Repetitive actions such as these contribute to process incrementation just as do other actions of the process, but additionally give the process a repetitive quality.

Some processes increment almost exclusively by repetition. Consider the familiar process for numerical integration depicted in Figure 3.1. The goal is to estimate the area under a curve in the range between *0* and *x* to a specified degree of accuracy (implied by the size of increment *dx*):



(1) Decide on a horizontal increment size *dx* that is appropriate to the desired accuracy of the estimate,

(2) Measure the height of the curve at 1*dx,

(3) Calculate the area of the rectangle described by 1*dx and the height of the curve,

(4) Add the area to a cumulative sum,

and repeat steps (2)-(4) for 2*dx, 3*dx,...(x/dx)*dx.

Figure 3.1. Numerical integration performed as a repetitive incremental process

The process primarily consists of a repetitive cycle in which steps (2) through (4) are repeated for a predetermined number of repetitions equal to *x/dx*. Each repetition serves to increment toward completion of the sought outcome (which is the area under the curve estimated to the specified degree of accuracy).

Are processes that are repetitive in this manner actually *iterative* in a meaningful sense? Perhaps. Although it is hard to imagine a design process that would, overall, be as orderly and predetermined as these purely repetitive processes are, it is conceivable that a design process could include specific tasks that are themselves purely repetitive. It therefore seems appropriate to acknowledge a purely repetitive pattern as a form of design iteration. The term *repetition iteration* will be used to refer to the specific pattern of process incrementation in which constituent actions take on a repetitive quality.

3.3 Progression Iteration

Most definitions of iteration suggest that it consists of something more than repetition alone. The definition cited above from Nukala et al. [1995] and Eppinger et al. [1997] associate iteration not only with repetition but also with the *improvement* or *evolution* of a design. Similarly, Urban and Hauser [1993] associate iteration with *refinement* of a design. These definitions seem to be referring to processes that achieve a succession of improvements, evolutions, or refinements on the way toward the final outcome, rather than a strictly incremental approach that achieves a single terminal outcome. These intermediate outcomes represent intermediate approximations of the goal that are achieved prior to the terminal outcome. This achievement of successive intermediate outcomes will be referred to here as *progression iteration*.

Many processes that could be performed purely incrementally could alternatively be performed progressively. Consider again the incremental numerical integration example. Its goal may alternatively be pursued in a progressive manner as depicted in Figure 3.2. Here, several intermediate area estimates are performed prior to the terminal one. Each intermediate outcome is completed by exactly the same repetitive algorithm as in the previous example, with only the element size and the necessary number of repetitions

being different. The key difference is that the process is now conducted in a way that achieves intermediate approximations along the way to the terminal approximation. These intermediate outcomes are less precise than the terminal outcome, but are available more quickly with less immediate expenditure than if the terminal outcome were pursued directly.

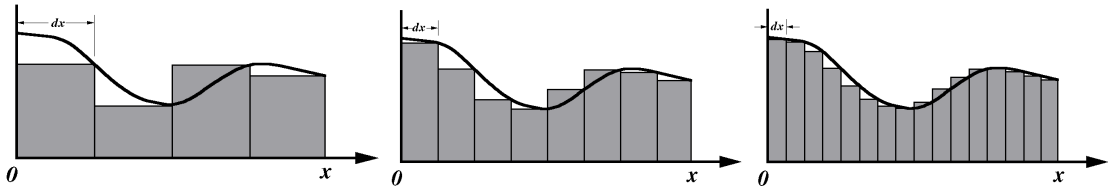
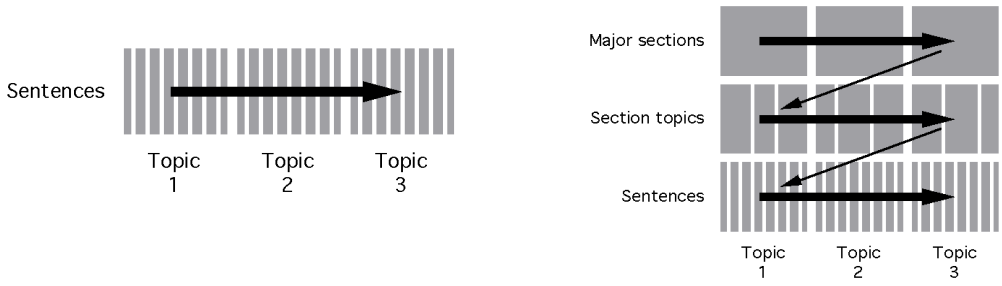


Figure 3.2. Numerical integration performed progressively

The same advantage is realized in the use of interlaced GIF (Graphic Interchange Format) graphics that are designed to display on a web page in several progressively detailed stages as they are loaded. The progressive display is meant to provide the viewer a rough sense of the entire image early in the loading process. Again, this progressive approach is an alternative to a purely incremental method (i.e., painting the image in full detail line by line) that would deliver the same end result.

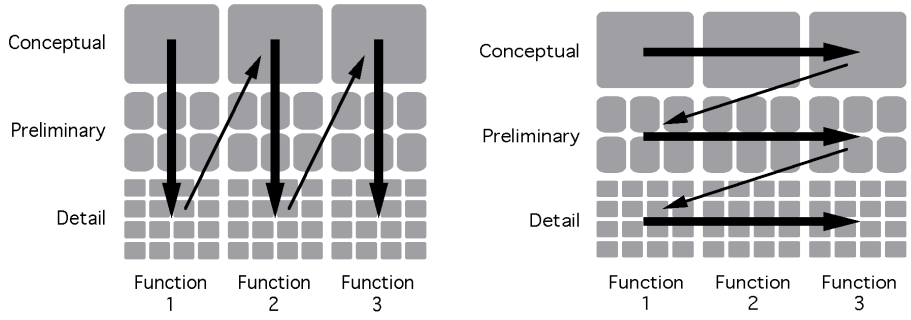
The choice between an incremental or progressive strategy also applies to more complex processes. For example, consider the problem of writing a rough draft of a report. An incremental strategy (Figure 3.3(a)) would have the writer begin with the opening sentence and repeat the writing of additional sentences in sequence until the entire report has been drafted. This process would simply increment a single, fully formed terminal outcome, and would likely be difficult and unsuccessful. A more agreeable strategy (Figure 3.3(b)) would instead have the writer outline the major sections of the entire report, and then outline the major topics of each section, and so on, gradually fleshing out the report in successive degrees of detail. Along the way, completion of each intermediate level of detail (e.g., the section topic outline for the entire report) is an intermediate outcome that represents an intermediate approximation of the goal.



(a) (b)

Figure 3.3. (a) Incremental and (b) progressive approaches to writing.

Design problems may also present choices between incremental and progressive strategies. The top-down, depth-first process of Figure 3.4(a), in which each required function of the design is successively conceived, laid out, and finalized, resembles a purely incremental approach because no intermediate approximations of the entire design (i.e. all three functions) are completed prior to the terminal outcome. The top-down, breadth-first approach of Figure 3.4(b) resembles a progressive approach, in which two intermediate approximations of the entire design are completed at the conceptual and preliminary levels prior to the final outcome at the detail level.



(a) (b)

Figure 3.4. (a) Incremental and (b) progressive approaches to design.

3.4 Contrasting Repetitive and Progressive Approaches

The numerical integration example demonstrates quite clearly that the decision to perform a process progressively rather than incrementally can increase the total number

of operations required to reach the terminal outcome (in this case, 28 rectangle areas must be calculated and summed instead of only 16). However, there are many situations in design activity where early, approximate outcomes delivered by a progressive approach can provide real advantages.

3.4.1 Task Complexity

Pursuit of intermediate outcomes can help maintain focus on the overall goal and impose organization on the task of selecting appropriate steps toward the goal. In the report-writing example, adopting a progressive approach rather than a purely incremental one allows the writer to revisit each topic several times at several levels of abstraction, focusing on overall organization at the higher levels and on rhetoric at the lower levels. If instead the goal were pursued purely incrementally, both of these considerations would have to be addressed concurrently, a task which is more difficult to perform effectively. Although it has been suggested that the choice of an incremental (i.e., top-down depth-first) approach may be related to a desire to minimize cognitive effort by focusing on distinct functional portions of the problem, a good design outcome tends to correlate better with the choice of a progressive (i.e. top-down breadth-first) approach [Gunther and Ehrlenspiel 1999].

3.4.2 Early Information Availability

Intermediate outcomes can make emergent information available to other portions of the project during early stages of design. The familiar "back of the envelope" calculation is a form of intermediate outcome, performed with a minimum expenditure of resources in order to provide approximate information that can be used temporarily until the inputs required for a more accurate estimate have stabilized. In the words of one industry executive, "we say it’s better to be 80 percent right fast than 100 percent right slow" [Wagoner 2002, p. 89]. For example, the shape of a vehicle wheel might be initially modeled as a cylindrical solid for the purpose of modeling weight distribution of the vehicle or for constructing a mockup, and later refined to a more realistic shape after options for tire size and hub construction have narrowed. Similarly, the progressive

approach to numerical integration provides a rough estimate of the curve area more quickly and at less immediate cost than would the purely incremental approach; and use of interlaced GIF graphics on a web page allows the viewer to make a judgement about the need to load the rest of the image well before it has finished loading.

3.4.3 Opportunity for Feedback

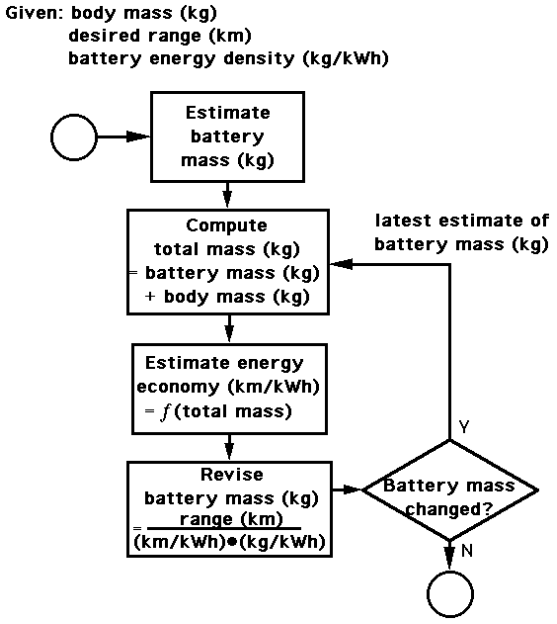
Perhaps the most significant advantage provided by the availability of intermediate outcomes is in the opportunity to evaluate their fitness relative to the goal. This information can then be used to direct subsequent steps of the process more effectively toward the goal. This type of information represents *feedback*. Feedback is defined here as information resulting from a test or evaluation of an intermediate outcome that returns information about its fitness relative to the goal.

Returning to the letter-sending example, a closer examination reveals that the process did not specifically include placing a postage stamp on the envelope. Unless a prepaid envelope was used, the outcome of the process would be a poor approximation of the goal, because the letter would soon be returned to the sender. At that point the goal may only be achieved by resuming the process and achieving a second outcome. Now, the first letter (without postage) may be understood as an intermediate outcome representing an intermediate approximation of the goal; the return of the letter represents feedback information about its fitness as an approximation of the goal; and the second letter (with postage) represents the second, terminal outcome. The process now possesses progression iteration that utilizes feedback about the fitness of a previous outcome.

One important use for feedback information is in testing for a stopping condition based on the result of the previous outcome. The letter-sending example above implicitly utilized a stopping condition, in that the process halts by default after the first outcome, unless and until feedback indicating nondelivery is received. In design, an obvious manifestation of feedback as a stopping condition is when the designer asks, "is it good enough yet?" or "does it work yet?" Even the progressive version of the numerical integration process of Figure 3.2 can be made to employ feedback as a stopping condition

by simply evaluating the magnitude of difference between the area estimates delivered by the previous and current outcomes. If the difference exceeds a specified tolerance representing the degree of accuracy desired, then the bar width *dx* is halved and another intermediate outcome completed; otherwise the latest outcome is accepted as the terminal outcome and the process terminates.

Sometimes, feedback information may also be employed by incorporating it directly into the generation of the next intermediate outcome, for example, to seed the next outcome. To illustrate, consider the problem of specifying a battery pack for an electric vehicle. The goal is to determine the necessary number of batteries (in terms of battery mass) required to meet a desired driving range (in kilometers). The necessary number of batteries depends on the total mass of the vehicle (heavier vehicles are less energy efficient), but the total mass depends in part on the number of batteries. The easiest way to solve this problem is with a numeric algorithm, as depicted in Figure 3.5.



Given:
Body mass (kg)
Desired range (km)
Battery energy density (kg/kWh)

(1) Provide seed estimate of battery mass (kg)

(2) Compute total vehicle mass:
= battery mass (kg) + body mass (kg)

(3) Compute energy economy (km/kWh):
= function of total vehicle mass

(4) Compute necessary battery mass (kg) at this energy economy:
= Range (km) / Energy economy (km/kWh)
* Energy density (kg/kWh)

(5) If new estimate of battery mass has changed significantly, repeat steps (2) - (4) using new estimate as seed; otherwise halt.

Figure 3.5. Repetitive and progressive numerical algorithm with feedback

In a repetitive cycle, an estimate of total mass (kg) is used to estimate the energy economy of the vehicle (km/kWh), which in turn is used to estimate the necessary battery

mass. The resulting estimate is used to update the total vehicle mass. After several repetitions the battery mass converges toward a fixed value and the process halts. This process possesses progression iteration because it repeatedly delivers increasingly accurate intermediate approximations of the necessary battery mass. It also employs feedback, which is represented by (a) the return of each intermediate approximation as a seed for the next approximation, and (b) the use of the difference between the last two approximations as a stopping condition.

3.5 Feedback Iteration

The concept of feedback is frequently encountered in conceptions of iteration. For example, some design process models liken the iterative loops that connect their various stages to a feedback control mechanism [Taguchi 1986], [Suh 1990], [Konda *et al.* 1992]. Iterative processes are commonly associated with the presence of feedback; for example, Braha and Maimon [1997] include feedback as an assumed property of an iterative model of design. It is well understood that feedback from testing a design can discover hidden rework [Cooper 1993], which is also associated with iteration. Dependency of tasks on feedback has also been shown to increase expected design time and number of iterations [Ahmadi and Wang 1994], and many negative outcomes related to poor feedback have been identified [Busby 1998].

This unique significance of feedback suggests that the presence of feedback in a progressive process deserves its own classification as a variety of iteration. Progression iteration that is guided by feedback regarding the fitness of intermediate outcomes shall be referred to as *feedback iteration*.

3.6 Selecting an "Iteration" to Measure

In summary, the preceding discussion has characterized processes as possessing a *goal* and a set of *constituent actions,* which act to complete an *outcome* that instantiates the goal. *Process incrementation* is the sequential performance of constituent actions that results in the incremental completion of an outcome. *Repetition* is a specific pattern by

which a process may increment toward an outcome; *progression* is a pattern of successive completion of intermediate outcomes; and *feedback* is an opportunity to evaluate an intermediate outcome generated in a progressive process. These latter three concepts are commonly associated with design iteration, leading to the recognition of three varieties of iteration: *repetition iteration, progression iteration*, and *feedback iteration.*

One is now faced with the question, which variety of iteration would be most valuable to measure in empirical research? Because the current study is specifically concerned with enabling research that would relate design iteration to design outcome, it is important to develop measures for a variety that is likely to be informative toward this goal. We will now consider several criteria that are important in this regard.

3.6.1 Criteria for Selection: Significance as an Independent Variable

Experimental research typically seeks to relate one or more independent variables to one or more dependent variables, in order to support or refute a hypothesis concerning a relationship between the variables. This suggests that the variety of iteration selected should be effective as an independent variable, which means that it should exhibit the following characteristics:

(1) Variability as an independent variable

Because experimental research is concerned with relating variations in an independent variable to variations in a dependent variable, it is important that the independent variable exhibit variation across likely study populations. Experimental studies that measure iteration as the independent variable would likely seek variations in iteration among process instances gathered from subjects performing the same design problem. Varieties of iteration that may be expected to exhibit significant variation in this context are more attractive as an independent variable than those that would be relatively invariant.

(2) Influence on dependent variables related to design outcome

The dependent variables that express the effect of an independent variable should be relevant to the goals of the experimental study that involves them. Because the current study specifically anticipates experimental studies leading to the development of design management guidelines, the dependent variable that holds the greatest interest is design outcome (i.e. design cost, design time, and design quality). Forms of iteration for which variations may be expected to cause variations in outcome are therefore of the greatest interest.

(3) Controllability in design practice

In order to have prescriptive value for design practice, the specific variations in the independent variable that lead to variations in the dependent variable should be controllable so that the effect may be controlled in practice. For example, forms of iteration that result from fundamental constraints such as the nature of a design problem are of less interest than those that result from decisions made by the designer.

The three varieties of iteration outlined above are now examined with respect to these three criteria.

3.6.2 How Significant Is Repetition Iteration?

Variability

In the previous examples of repetitive processes, their repetitive aspect came about as a result of either the problem definition or of constraints acting on its solution. For example, in the numerical integration process, the fact that repetitions are necessary at all stems from the variable shape of the curve. Because the curve varies dramatically in height and slope, only a relatively large number of relatively narrow rectangles can deliver an accurate area estimate, leading to an inherently repetitive solution process. Even the necessary number of repetitions is fixed because it depends on the size of the increment *dx,* which is a result of the desired accuracy of the terminal outcome. These factors are determined by the problem statement and so are not in control of the person

carrying out the process; therefore one would not expect much variation in repetitive character if the process were performed multiple times by different individuals. However, if the person carrying out the process were able to adjust the problem statement to modify the necessary repetitive aspect of its solution, some variability in repetitive character could result.

Influence

In general, the cost and duration of a repetitive process may be expected to increase with the number of repetitions necessary to complete it, because one may generally expect some degree of overhead cost, such as setup time, to be associated with each repetition. This alone suggests a reason to expect variations in repetition iteration to bring about variations in the cost and time aspect of design outcome. It seems less likely that variations in the number of repetitions would lead to large differences in the quality of the outcome, owing to the generally predetermined source of the repetitive aspect of such processes.

Controllability

In the letter-writing example, the envelope-sealing subprocess was said to possess a potential repetitive quality in its successive applications of pressure to the seal. Similarly, the coffee-making example noted that the addition of coffee into the filter might call for repetitive scooping actions. In both of these cases, these repetitive actions result from capacity constraints. The need to apply more than one application of pressure to the seal is largely related to the limited size and strength of one's fingers and the inability to slide pressure across both edges of the flap simultaneously when only two hands are available. The need to perform repetitive scoops of coffee arises when the capacity of the scoop is less than the amount of coffee needed. Capacity constraints that result in repetitive actions may potentially be alleviated by a more appropriate allocation of resources to these tasks, resulting in a reduction in the necessary number of repetitions. Where resource allocation is within the control of the designer, this type of capacity-related repetition may be controllable in practice.

3.6.3 How Significant Is Progression Iteration?

Variability

The observation that designers rarely stick to a perfectly depth-first (incremental) or perfectly breadth-first (progressive) process indicates a relatively strong potential for individual variability in progression iteration. Fricke [1996] found that two general design strategies emerged, similar to breadth-first and depth-first. In what was called a stepwise, process-oriented strategy, designers worked on different problem areas (e.g., different components) at one level of abstraction before proceeding to the next level of abstraction. In the alternative "function-oriented" strategy, designers tended to work within one problem area from abstract to concrete levels of abstraction, then proceeded to the next component. All of the designers followed a different combination of breadth-first and depth-first strategies at different times during the process. Because choosing progression over incrementation also provides benefit in terms of process organization and a reduction in complexity, it suggests a relation to the designer's capacity to handle complexity. These factors suggest that variations in progression iteration could be expected among individual designers.

Variations in preference between a progressive or incremental strategy may also emerge from cognitive constraints encountered by the individual. Recalling the previous examples of progressive report writing and progressive numerical integration, they were preferable to their purely incremental counterparts because they reduce the complexity and difficulty of orienting the process toward an acceptable goal. This implies that different individuals will find it necessary to rely on this strategy to differing degrees.

Influence

The general expectation that a progressive process can reduce complexity and difficulty directly suggests that variations in the degree to which it is employed may lead to variations in outcome.

Controllability

In demonstrating that many processes may be executed in either an incremental or progressive manner, the previous examples illustrate that the decision is largely within control of the designer, meaning that prescriptive guidelines relating to progression iteration should be possible to implement if developed.

3.6.4 How Significant Is Feedback Iteration?

Variability

Feedback iteration, being a form of progression iteration, can be expected to have similar implications for variability. The added dimension of feedback that accompanies each progression provides an additional means of variation. For instance, processes may not only vary in the degree to which tehy employ feedback iteration, but also in the specific forms of feedback they generate.

Influence

The suggestion that variations in reliance on feedback are likely to cause variations in outcome has a particularly strong argument. Feedback carries significant implications for the effectiveness of design planning, which has a potentially strong influence on outcome. For example, consider that in a non-feedback cycle, such as that of repetition iteration or progression iteration, all of the information employed in each constituent action is initially available in the problem statement. That is, all rectangle area calculations were based upon the height of the curve at various points (given), and the increment *dx* (given or computed deterministically using given information). This suggests that every necessary step could be be predicted in advance so that appropriate resources may be allocated before the process even begins. Feedback information, however, is unique in that it represents information embodied in the evolving state of the design, rather than information initially accessible in the problem statement. Processes involving feedback information are therefore much more difficult to plan. In particular, when feedback information is necessary to seed or otherwise inform subsequent intermediate outcomes, it becomes very difficult to anticipate the number and nature of

the constituent actions that will eventually take place before the process halts. The process is therefore susceptible to unpredictable conditions that can inflate cost and time beyond that budgeted or result in a reduction in quality due to premature consumption of budgeted resources.

Another important result of the involvement of feedback information is the potential for *divergent* rather than convergent progression. In a purely progressive process (one that does not evaluate intermediate outcomes via feedback), there is no reason to expect intermediate outcomes to diverge away from the terminal outcome because the goal is known *a priori* and all information applicable to the process that pursues it is known at the outset. Without feedback, there is no source for information that could cause the process to diverge from the original intent. Feedback, however, could lead to actions that would otherwise not have a reason to occur, such as the undoing of previous work or the consideration of changes to the problem statement. Subsequent outcomes could diverge, potentially leading to a radically different outcome. This suggests that variations in feedback iteration could be responsible for particularly significant variations in outcome.

Controllability

Can reliance on feedback iteration be controlled by decisions of the designer? A request for feedback information may be understood as representing the designer's choice of testing over analysis. From a deterministic perspective, the fitness of a design may theoretically be derived analytically by applying first principles to determine the performance of the current design and comparing it to the goal statement. Frequently, however, analysis is not the most practical way to obtain this information. Especially in later stages of the design process when the design has become complex, testing the current design for fitness is often a more practical alternative than to continue with an increasingly complex and uncertain analysis. This tradeoff represents a decision that a designer is capable of recognizing, evaluating, and acting upon if the proper guidelines are available.

3.7 Conclusion: Develop Measures for Feedback Iteration

The general conclusions of this discussion are summarized in Table 3-1.

Table 3-1. Comparison of effectiveness as an independent variable

	Variability	**Influence**	**Control**
Repetition Iteration	weak	moderate	weak
Progression Iteration	moderate	moderate	moderate
Feedback Iteration	strong	strong	strong

Repetition iteration shows some potential for variability, influence, and controllability, but these effects are weak to moderate compared to those of the other varieties. Progression iteration has a potentially stronger degree of interest, but feedback iteration is particularly strong. The current study therefore seeks to develop measures of feedback iteration.

4 Measures of Iteration

The previous chapter identified several varieties of iteration and distinguished among them in terms of their potential interest to the goal of empirically relating design iteration to design outcome. *Feedback iteration* was judged as holding particular interest and was selected as a subject for the development of measures. This chapter describes an approach to developing such measures. It concludes with the development of several measures that are then applied and evaluated in subsequent chapters.

An Approach to the Objective Measurement of Qualitative Data

Analyzing qualitative data such as that gathered from the observation of design activity can be difficult, and often calls for subjective interpretation [Chi 1997]. Improving the objectivity of interpretation calls for measures that yield a similar measurement result regardless of who applies them.

One approach to the application of measures to design activity data consists of constructing a representation of the data and applying an appropriate set of metrics to the representation. For example, in research that involves timeline representation of the design process, it is not uncommon to witness the application of simple metrics to the timeline depictions, such as number of transitions from one design step to another, or the total duration of the process. Such metrics may serve as base quantities upon which derived quantities that are more informative may be constructed. For example, the total number of transitions might be divided by the time duration of the process to yield an average transition rate, which might suggest something about the pace of one designer relative to that of another [Atman et al. 1999].

A metric-based measurement strategy such as this calls for an appropriate means to represent an observed design process, and an appropriate set of metrics to apply to it. Our strategy will consist of the representation of design processes in the form of timelines, and the measurement of design process variables that the timelines highlight.

Measures of iterative character will then be constructed from these base measures. The remaining sections of this chapter develop the primary components of this strategy:

(1) **A model of design** on which timeline representation formats may be based;

(2) **Schemas for timeline representation** that are suggested by the model;

(3) **A set of base metrics** derived from specific patterns of activity that can be portrayed under the timeline schemas; and

(4) **Measures of iterative character** derived from these metrics.

4.1 A Model of Design

The various timeline representations that were reviewed in Chapter 2 employ a variety of event categorizations as the vertical axis of the timeline. In each case, some model of design activity underlies the selection of these categories. For example, to categorize observed activities according to stages of design such as problem definition, information gathering, and so on [Atman and Bursic 1998, Atman *et al*. 1999] is to base the timeline on a traditional model in which design is conceived as being composed of activities that belong to a group of distinct, generally sequential stages. The choice of a specific design model as a basis for a timeline representation is likely to influence what the timeline is capable of expressing, and this decision is likely to be influenced by the specific goals of the study.

In order to develop a timeline representation that is likely to lead to the measurement of feedback iteration, a first step is to adopt a model of design that specifically accounts for the presence of feedback in design activity. Recalling the examples of feedback described in the previous chapter, it is apparent that feedback is a form of information, specifically concerning the fulfillment of functional requirements relating to the goal of the design process. Adopting an information processing perspective on design activity is a natural way to begin searching for a design model that accounts for feedback.

4.1.1 An Information-Processing Model of Design Activity

Design is commonly understood and modeled as an activity of information processing [e.g. Wallace and Hales [1987], Ullman et al. [1988], Safoutin [1990], Safoutin and Thurston [1993]. These models hold that the information processing aspect of design activity is key to understanding design at its most basic level. For example, one model of design that adopts this perspective is the Task Episode Accumulation (TEA) model [Ullman et al. 1988]. The TEA model teaches a conceptualization of design as a process dominated by information application and retrieval, in which the designer employs various information operators that work to advance the design by the application and evaluation of information that surrounds it. In very simple terms, under this model a design artifact grows by the accumulation of applied information, and information that thereby embodies the artifact is periodically retrieved for the purpose of verification and evaluation. Specific operators such as the "Assimilate" operator acknowledge various aspects of information application, and others such as the "Evaluate" operator are directly related to the generation and application of feedback information.

With respect to the current study, the value of this and other information processing perspectives of design activity is in their recognition of the phenonema of information application and information feedback. This allows the recognition of a very simple but profound distinction between two sources of applied information - from the design environment, which according to Ullman consists of sources such as domain knowledge, reference manuals, training, education, and experience; and from the information content of the design, in terms of feedback about its fitness.

In this study, the term *feedback* refers to information retrieved from the design state, specifically regarding its fitness with respect to the functional requirements for which the artifact is being designed. The counterpart to feedback is *assimilation*, which refers to the application of information from any source that cannot be identified as feedback.

In a design process there are often many opportunities to generate information that qualifies as feedback. For example, one may simply perform a mental simulation of the interaction of two hypothesized components to judge their compatibility; or one may

construct a highly accurate prototype of a component or an entire design embodiment and test it in an environment similar to that in which it is expected to be used. Although various forms of feedback may vary dramatically in their accuracy, cost, and significance, they qualify as feedback because they represent information concerning the fitness of the design, either in the form of an approximation of the entire design or a portion of it.

4.1.2 A Parametric Model for the Design Task

We now turn toward selecting a model for the design problem itself. Although design problems are said to share attributes that distinguish them from non-design problems [Goel 1994], a number of distinct classes of design problem have been recognized. Some examples are routine design, conceptual design, parametric design, and catalog design [Ullman 1989]. To highlight assimilation and feedback in an empirical setting, there are advantages to adopting a *parametric* model.

Parametric design has been described as "the design problem of refining from an artifact *type* to a *specific* artifact" [Ullman 1989]. Structuring a design task parametrically can facilitate understanding of the structure of the problem and can support a systematic approach to its solution. A properly structured parametric design problem models the artifact type as a set of distinct *design parameters* (DPs) that may be mapped to one or more specific *functional requirements* (FRs). Design alternatives may then be expressed in terms of parameter values, and their corresponding functional performance is understood to be a function of these values. Such a parametric structure is central to the axiomatic approach to design advocated by Suh [1990], which requires that one represent the design problem as a transform between distinct DPs and FRs, as shown in Figure 4.1. In Suh's application, the parametric model serves to characterize the overall nature of the designer's work as one of discovering a satisfactory transform matrix consisting of specific parameter values (pv). It also provides a formal construct that leads to hypotheses regarding the quality of various classes of parametric solutions. Although design problems vary in the degree to which distinct DPs and FRs are initially apparent, Suh's formulations implicitly assert that a parametric specification is applicable to virtually any design problem.

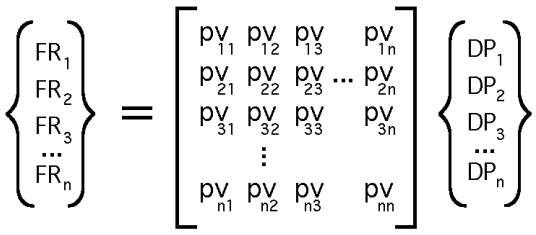


Figure 4.1. Parametric model of the design problem [Suh 1990].

For our purposes, the value of adopting a parametric model is in providing a place to look for activity that represents assimilation and feedback. Assimilation becomes associated with DPs, and consists of the specification of parameter values. Feedback becomes associated with the evaluation of specific FRs. Recognition and capture of assimilation and feedback in an empirical setting may then be focused upon these specific designer activities.

Summary: A Parametric, Information Processing Model of Design

Our information-processing, parametric model of design has described design as a series of information processing events consisting of *assimilation* and *feedback*, which take place with respect to a parametrically structured design task defined by *design parameters* and *functional requirements.* These variables will be referred to by the labels *A*, *F*, *DP*, and *FR*, respectively. The model is depicted in Figure 4.2.

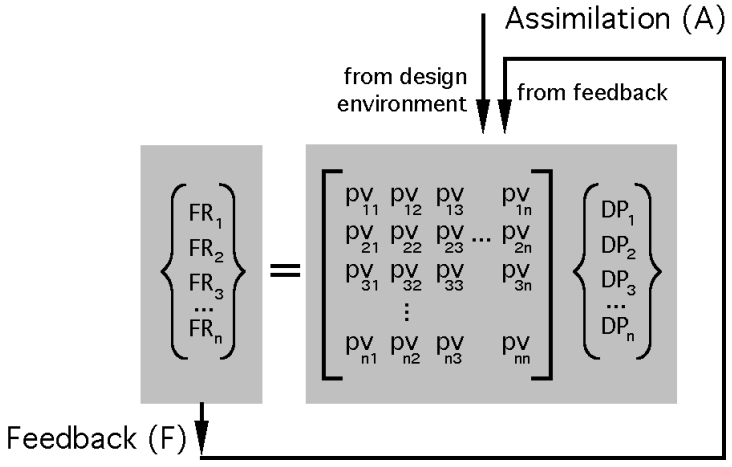


Figure 4.2. Information processing, parametric model of design

4.2 Timeline Schemas

We now turn toward the development of schemas for timeline representation of design processes under the parametric, information-processing model of design. Given a specific model of design, the problem of depicting design activity on a timeline consists of constructing a representation space based upon the variables supplied by the model, and plotting these variables over time. Various combinations of the four major variables *A*, *F*, *DP*, and *FR* lead to several potential timeline formats.

4.2.1 Assimilation-Feedback (*A*-*F*) Timeline

One potential format results from simply plotting assimilation (A) and feedback (F) events over time, without regard to the design parameters or functional requirements they are associated with. This creates what will be called an *A*-*F timeline*. Figure 4.3 depicts an example of this timeline format, in which assimilation events are depicted as rectangles and feedback as diamonds:

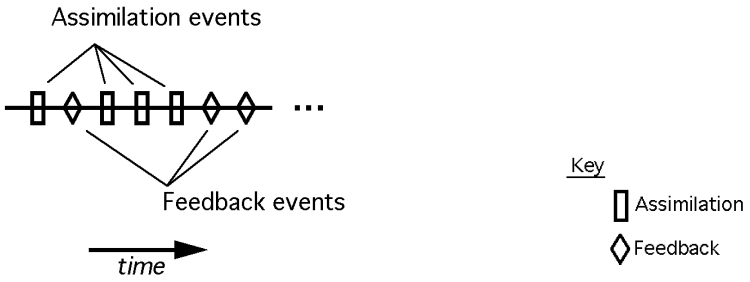


Figure 4.3. Example *A*-*F* timeline

The *A*-*F* schema is limited to depicting sequential patterns in these events. Because events occur in only two types, *A* and *F*, the pattern may either consist of a succession of a single type or a switch from one type to another. An *episode* will be defined as any set of sequential events of the same type. Thus an *assimilation episode* consists of one or more individual assimilation events that are bounded by two successive feedback events. Similarly, a *feedback episode* consists of one or more feedback events bounded by two

successive assimilation events. An example showing several assimilation and feedback episodes is provided in Figure 4.4.

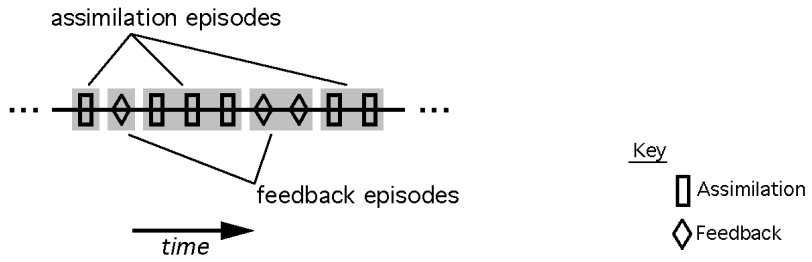


Figure 4.4. Assimilation and feedback episodes in an *A-F* timeline

The significance of a multi-event assimilation episode lies in the suggestion that downstream design parameter values in the episode are being set in the absence of feedback about the effect of the upstream settings. That is, the second parameter is set without evaluating the effect of the first parameter setting. If each parameter in the episode is known to be functionally independent, that is, if their only influence is on separate functional requirements, then this may not be a significant problem, but often this is not the case. Because feedback regarding the effect of the upstream parameter has not been generated, the designer must be basing downstream parameter settings using information other than feedback, such as preexisting knowledge and experience (or is simply making a guess).

Similarly, a multiple-event feedback episode indicates that the designer is repeatedly requesting feedback without having made any intervening changes to the design. This suggests that more than one functional requirement is being evaluated with respect to previous parameter settings. For example, in order to evaluate the effect of a given material specification on total cost of the product, one might first request feedback information about the total material cost, followed by feedback about projected manufacturing cost. A large number of events in a feedback episode suggest that the designer is evaluating a broad range of FRs with respect to a fixed design state, which suggests that the current state is being taken rather seriously as a potential solution.

In this study, the primary purpose of discussing the *A-F* timeline is to formalize the concept of assimilation and feedback episodes. Although the *A-F* timeline format was not selected for application in this study, its implications for measurement of iteration are examined in detail in Appendix D.

4.2.2 A-F-DP and A-F-FR Timelines

Adding a second, vertical dimension to the timeline format allows one additional design model variable to be depicted: either the DPs associated with each A, or the FRs associated with each F (but not both). This would result in what could be called *A-F-DP* and *A-F-FR* formats. Similarly, adding a third dimension would result in an *A-F-FR-DP* timeline. This would require a three-dimensional depiction. Traditionally, most timelines have taken on a two-dimensional format in a compromise between expressiveness and practicality. The possibility of an *A-F-FR-DP* format is acknowledged but will be reserved for future investigation; the remainder of this discussion is restricted to two-dimensional formats.

In an *A-F-DP timeline* schema, DPs are depicted in the vertical dimension of the timeline, as depicted in Figure 4.5. This allows each assimilation event to not only be plotted in time but also to be positioned vertically to indicate the specific design parameter it applies to. Feedback events, however, are not explicitly associated with design parameters and so they still may only be plotted in time, as a vertical line. Figure 4.5 also depicts the appearance of assimilation and feedback episodes in this timeline format.

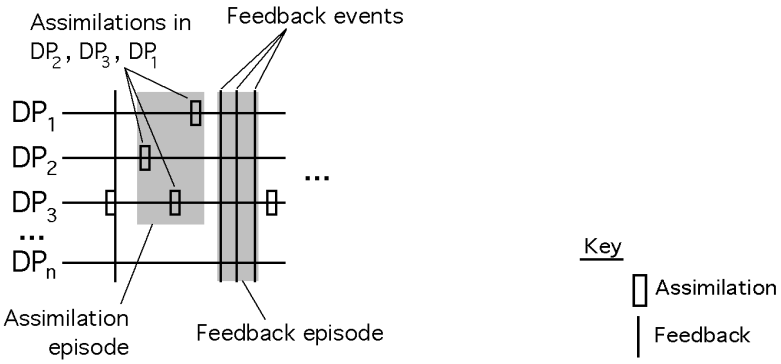


Figure 4.5. Example *A-F-DP* timeline

Alternatively, in an *A-F-FR timeline* schema, FRs are depicted in the vertical dimension, as illustrated in Figure 4.6. This allows each feedback event to be positioned vertically to indicate the specific functional requirement it evaluates. Assimilation events can only be plotted as a vertical line.

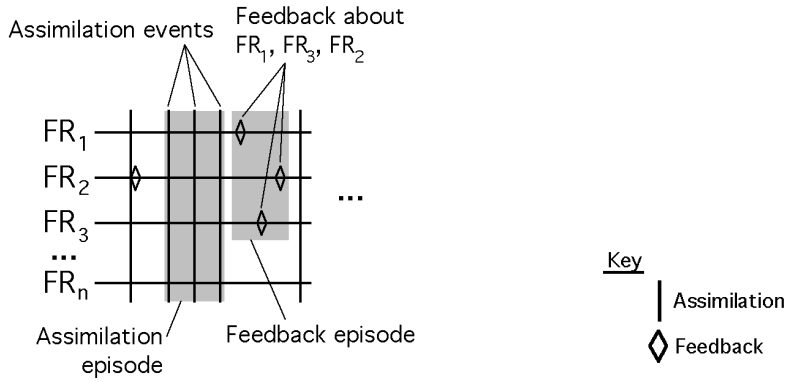


Figure 4.6. Example *A-F-FR* timeline

These timeline formats will now be systematically examined in terms of the patterns of episodes that they are capable of depicting. Several specific patterns that are particularly suggestive of relevant base metrics will be examined in detail.

4.3 Patterns and Base Metrics

The *A-F-DP* and *A-F-FR* timeline schemas provide two distinct frameworks for the depiction of assimilation and feedback events. The manner in which events are depicted under each schema leads to a set of characteristic patterns that are endemic to the schema. These patterns provide a possible basis for metrics by which a depicted design process might be measured.

The current study examines patterns that arise from consideration of a sequence of three sequential episodes of assimilation or feedback, which will be referred to as a *triplet*. The triplet approach is limited to revealing patterns of a highly localized sort that can be expressed in a group of three sequential episodes. Other patterns of iteration are likely to exist that this approach would not identify. For example, a human looking at a

timeline can easily detect the possible existence of patterns that exist on a broader scale, such as repeated visitations to specific parameters or groups of parameters over the course of the process. Systematic identification of patterns such as these would call for a different approach. Despite its limited scope, the triplet approach does reveal patterns that are potentially informative to design outcome and that are measurable in an empirical setting. The following sections explore this approach by systematically examining all possible triplet patterns that can be recognized under the *A-F-DP* and *A-F-FR* timeline schemas.

4.3.1 Timeline Information Attributes

Information Attributes of the A-F-DP Timeline

In an *A-F-DP* timeline, let us consider a sequence consisting of a feedback episode (F) bounded by two assimilation episodes (A). This *A-F-A* sequence contains the most information compared to the alternative *F-A-F* sequence, because it contains two *A* events which carry additional *DP* information. In an *A-F-DP* timeline, one is thus concerned with discerning patterns in *A* across a given *F*.

Three types of information about each *A* episode are carried in an *A-F-DP* timeline: the fact that the episode *occurred* (denoted by the notation a), the *quantity* of design parameters that were edited in the episode (denoted by n_{DP}), and the specific *identity* of these parameters (id_{DP}). This can be seen by referring again to Figure 4.5, in which may easily discern that the episode occurred, that it consisted of specifically three events, and that DP1, DP2, and DP3 were affected. With regard the the *F* episode, only two types of information are carried: the fact that the episode *occurred* (denoted as f), and the *quantity* of FRs that it evaluated (n_{FR}). This again can be seen in Figure 4.5, in which it can be established that a feedback episode occurred and that it consists of specifically three events. However, information about the FRs associated with these events is not represented. Table 4-1 summarizes the types of information that are therefore accessible in any *A-F-A* sequence in an *A-F-DP* timeline.

Table 4-1. Accessible information in an *A-F-A* sequence

episode *i-1 (A)*	episode *i (F)*	episode *i+1 (A)*
Occurrence (*a*) DP quantity (n_{DP}) DP identity (id_{DP})	Occurrence (*f*) FR quantity (n_{FR})	Occurrence (*a*) DP quantity (n_{DP}) DP identity (id_{DP})

An A-F-A sequence therefore can be translated to a large variety of specific triplets and doublets, consisting of various combinations of the quantities shown in the table. Each combination represents a specific *pattern* that may be extracted. For example, the triplet

$$(n_{DP}) : (f) : (n_{DP})$$

represents the pattern that is being discerned when one considers the number of parameters (n_{DP}) that were edited prior to a feedback episode (*f*), and compares it to the number of parameters were edited prior to the next feedback episode (n_{DP}). Alternatively, the doublet pattern

$$(n_{DP}) : (f)$$

represents consideration of only the number of parameters edited (n_{DP}) prior to a single feedback episode (*f*).

Information Attributes of the A-F-FR Timeline

Next, let us consider the information attributes of an *A-F-FR* timeline. Here, an *F-A-F* sequence contains more information than the alternative *A-F-A* sequence, because *F* events carry additional *FR* information. In an *A-F-FR* timeline, one is thus concerned with patterns in *F* episodes across *A* episodes.

Three types of information are accessible about each *F* episode: the fact that the episode *occurred*, the *quantity* of FRs that were evaluated in the episode, and the specific *identity* of these FRs. Only two types of information regarding the assimilation episode

are accessible: the fact that the episode *occurred*, and the *quantity* of DPs that it affected. Again, this may be confirmed by examining the example timeline of Figure 4.6. These information possibilities are depicted in Table 4-2.

Table 4-2. Accessible information in an *F-A-F* sequence

episode *i-1 (F)*	episode *i (A)*	episode *i+1 (F)*
Occurrence (f) FR quantity (n_{FR}) FR identity (id_{FR})	Occurrence (a) DP quantity (n_{DP})	Occurrence (f) FR quantity (n_{FR}) FR identity (id_{FR})

The A-F-FR timeline thus leads to an additional set of specific triplets and doublets that suggest specific patterns. The next section examines these patterns in detail.

4.3.2 Episode Patterns

Enumerating each triplet/doublet pattern that is possible under the *A-F-DP* and *A-F-FR* timeline schemas yields a large collection of potential episode patterns that might be sought in either type of timeline. The number of combinatorial possibilities is reduced substantially after accounting for redundance in the *occurrence* variable. When applied to both bounding episodes (the first or last position in a triplet), this variable creates only (*a*):(*f*):(*a*) and (*f*):(*a*):(*f*) triplets. These triplets are not significant because, by definition, all processes must be composed of an alternating sequence of *a* and *f* episodes (that is, every feedback episode is by definition bounded by two assimilation episodes, and vice versa). After eliminating these possibilities, sixteen possible patterns emerge for each of the two timeline schemas.

For an *A-F-DP* timeline, Table 4-3 depicts each potential pattern and provides a verbal interpretation of the situation it implies. Table 4-4 similarly depicts each potential pattern for an *A-F-FR* timeline.

Table 4-3. Episode patterns in an A-F-DP timeline

	episode *i-1* *(A)*	episode *i* *(F)*	episode *i+1* *(A)*	Interpretation	Pattern name
1a	**n_{DP}**	**f**	**-**	**Feedback is preceded by revision of a given quantity of DPs**	**Parameter quantity**
2a	-	f	n_{DP}	Feedback is followed by revision of a given quantity of DPs	-
3a	n_{DP}	f	n_{DP}	Successive feedback episodes concern revisions to same or different quantity of DPs	Parameter breadth
4a	id_{DP}	f	-	Revision of specific DPs is followed by feedback	-
5a	-	f	id_{DP}	Feedback is followed by revision of specific DPs	-
6a	**id_{DP}**	**f**	**id_{DP}**	**Successive feedback episodes concern revisions to same or different specific DPs**	**Parameter identity**
7a	n_{DP}	f	id_{DP}	Feedback concerning revisions to a given quantity of DPs is followed by revision of a specific set of DPs	-
8a	id_{DP}	f	n_{DP}	Feedback concerning revisions to a specific set of DPs is followed by revisions to a given quantity of DPs	-
9a	n_{DP}	n_{FR}	-	Revision of a given quantity of DPs is followed by feedback about a given quantity of FRs	-
10a	-	n_{FR}	n_{DP}	Feedback about given quantity of FRs is followed by revision of a given quantity of DPs	-
11a	n_{DP}	n_{FR}	n_{DP}	Successive feedback episodes about a same or different quantity of FRs concern a same or different quantity of DPs	-
12a	id_{DP}	n_{FR}	-	Revision of specific DPs is followed by feedback about a given quantity of FRs	-
13a	-	n_{FR}	id_{DP}	Feedback about a given quantity of FRs is followed by revision of specific DPs	-
14a	id_{DP}	n_{FR}	id_{DP}	Successive feedback episodes concern revisions to same or different specific DPs	-
15a	n_{DP}	n_{FR}	id_{DP}	Feedback about given quantity of FRs, concerning revisions to a given quantity of DPs, is followed by revision of specific DPs	-
16a	id_{DP}	n_{FR}	n_{DP}	Feedback about given quantity of FRs, concerning revisions to specific DPs, is followed by revisions to a given quantity of DPs	-

Table 4-4. Episode patterns in an A-F-FR timeline

	episode *i-1* *(F)*	episode *i* *(A)*	episode *i+1* *(F)*	Interpretation	Label
1b	n_{FR}	a	-	Assimilation is preceded by evaluation of a given quantity of FRs	-
2b	-	a	n_{FR}	Assimilation is followed by evaluation of a given quantity of FRs	-
3b	n_{FR}	a	n_{FR}	Successive assimilation episodes concern evaluations of same or different quantity of FRs	FR breadth
4b	id_{FR}	a	-	Evaluation of specific FRs is followed by assimilation	-
5b	-	a	id_{FR}	Assimilation is followed by evaluation of specific FRs	-
6b	id_{FR}	a	id_{FR}	Successive assimilation episodes concern evaluations of same or different specific FRs	FR identity
7b	n_{FR}	a	id_{FR}	Assimilation following evaluations of a given quantity of FRs is followed by evaluation of specific FRs	-
8b	id_{FR}	a	n_{FR}	Assimilation following evaluations of specific FRs is followed by evaluation of a given quantity of FRs	-
9b	n_{FR}	n_{DP}	-	Evaluation of a given quantity of FRs is followed by revision of a given quantity of DPs	-
10b	-	n_{DP}	n_{FR}	Revision of a given quantity of DPs is followed by evaluation of a given quantity of FRs	-
11b	n_{FR}	n_{DP}	n_{FR}	Successive assimilation episodes affecting a same or different quantity of DPs are evaluated in terms of a same or different quantity of FRs	-
12b	id_{FR}	n_{DP}	-	Evaluation of specific FRs is followed by revision of a given quantity of DPs	-
13b	-	n_{DP}	id_{FR}	Revision of a given quantity of DPs is followed by evaluation of specific FRs	-
14b	id_{FR}	n_{DP}	id_{FR}	Successive assimilation episodes concern evaluations of same or different specific FRs	-
15b	n_{FR}	n_{DP}	id_{FR}	Revisions to given quantity of DPs, following evaluation of a given quantity of FRs, is followed by evaluation of specific FRs	-
16b	id_{FR}	n_{DP}	n_{FR}	Assimilation in a given quantity of DPs, following evaluations of specific FRs, is followed by evaluations of a given quantity of FRs	-

4.3.3 Selecting Patterns of Interest

The purpose of seeking a pattern in an observed design process would be based upon an expectation that the pattern may be significant to some question of interest. Most of the patterns depicted in the tables do not have obvious significance. However, a few of the patterns do have a concrete interpretation, and some of these interpretations have implications for design outcome. These patterns will be discussed in the following sections.

Parameter Quantity Pattern

Episode pattern 1a (Table 4-3) is particularly interesting. It describes the quantity of parameters that are edited prior to a feedback event, and will be referred to as the *parameter quantity pattern.*

As we have already established, a feedback event returns information regarding the fulfillment of a functional requirement. When the designer changes parameter values and seeks feedback about the effect, it represents an opportunity to learn the effect of specific parameter changes on design performance so that this information may be applied to subsequent decisions.

In this regard, the most reliable information is generated when the observed effect can be isolated to a single variable. Multiple variables act to confound the effect and so the observed effect cannot provide as reliable a lesson. This suggests that variations in the parameter quantity pattern might be relatable to variations in design outcome. A designer who habitually edits multiple parameters before requesting feedback generates information that is less informative than does a designer who habitually concentrates each feedback request on a single parameter. This suggests that a desirable outcome will be more difficult to find. For this reason it is potentially attractive to distinguish among observed processes in terms of the parameter quantity pattern.

Identifying the parameter quantity pattern in an A-F-DP timeline is straightforward. Figure 4.7 shows an unconfounded feedback event in which only one parameter is active.

In contrast, Figure 4.8 shows a feedback event prior to which multiple parameter values have been changed. By identifying these events, a design process might be characterized in terms of its relative proportion of each type of event.

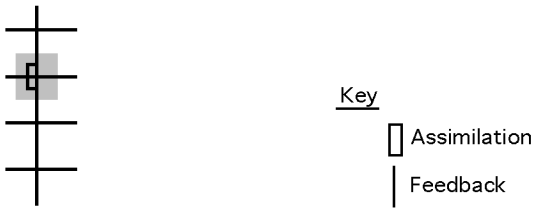


Figure 4.7. Unconfounded parameter quantity pattern

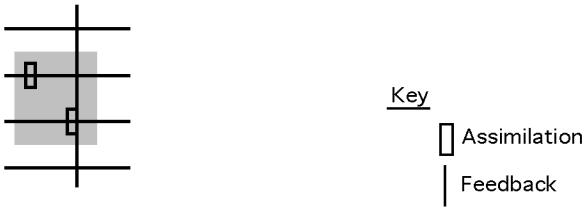


Figure 4.8. Confounded parameter quantity pattern

Alteratively, rather than counting events based on whether they respond to one parameter change or more than one, it is also possible to base the metric on the average number of variables that confound a typical feedback event. This computed application of the metric is developed further in Appendix E.

Parameter Identity Pattern

Episode pattern 6a (Table 4-3) is another interesting pattern. It compares the *identity* of parameters between two successive feedback events. That is, it indicates whether the designer is submitting the same set of parameters repeatedly or is switching to different parameters. The pattern that describes this issue will be referred to as *parameter identity.* The total number of parameters is not important to this pattern; each set could consist of a similar number or a different number.

Like the parameter quantity pattern, the parameter identity pattern is also suggestive of an influence on design outcome. Repetition of feedback requests that involve a

stationary set of parameters suggests a focused optimization cycle, in which knowledge is cumulatively being built about the influence of this set of parameters on design performance, and is immediately being reapplied to the revision of the same parameters. In contrast, to frequently travel from one set of parameters to a different set interrupts the learning process about the first parameter set, suggests that the information returned is not being immediately applied to decisions about the same parameters, and allows the information to be confounded by the effect of intervening changes to other parameters that take place in the meantime. By this reasoning a process characterized by frequent travel from one set of parameters to a different set potentially generates and applies feedback information less effectively than one that remains stationary across feedback events more often.

For this reason it is potentially attractive to distinguish among observed processes in terms of the parameter identity pattern. Figure 4.9 depicts a stationary parameter identity pattern in which successive feedback episodes encounter the same set of parameters. In Figure 4.9(a), changes to the same single parameter are the subjects of both successive feedback events. In Figure 4.9(b), the same set of two parameters is the subject of each event.

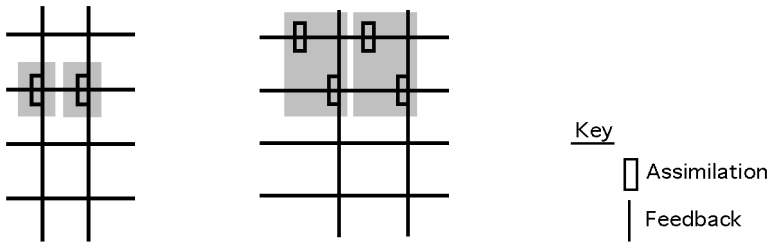
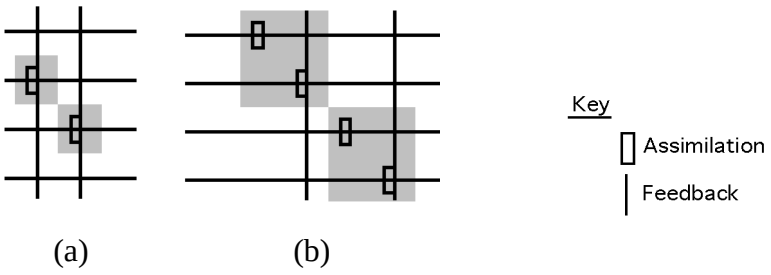


Figure 4.9. Stationary parameter identity patterns

In contrast, Figure 4.10 depicts traveling patterns in which a different parameter set is edited across successive feedback events. In these examples, none of the parameters in the first event are present in the second.



(a) (b)

Figure 4.10. Traveling parameter identity patterns

It follows that a confounded feedback pattern (one that submits multiple parameters) may be partly stationary and partly traveling, if some parameters are continued in the next feedback event while others are not. This situation can be accounted for by calculating the proportion of parameters that continue. For example, if the first feedback event involves DP1, DP3, and DP5 while the second involves DP1, DP2, DP3, and DP7, then 2/3 of the parameters are continued and 1/3 are not. (Incidentally, the appearance of DP2 and DP7 in the second episode is inconsequential to assessing the parameter identity pattern because neither is present in the first episode and therefore cannot be regarded as having been continued).

Other Patterns

A number of other patterns have interpretations that are conceptually meaningful, although they do not have immediately obvious implications for design outcome. These patterns are discussed below for completeness.

Episode pattern 3a (Table 4-3) concerns the relative number of parameters that are active in two successive feedback events. That is, it indicates whether the designer is submitting an increasing, decreasing, or constant number of parameters to the next feedback episode. This pattern might be referred to as *parameter breadth*. The identity of the parameters is disregarded; both sets could consist of similar parameters or be entirely different. However, an expanding number of parameters guarantees that new parameters have appeared, and the parameter focus is widening; a decreasing number guarantees that the focus is narrowing.

Episode pattern 3b (Table 4-4) represents a breadth pattern similar to parameter breadth, but with respect to functional requirements instead of design parameters. This pattern might be referred to as *FR breadth*. It concerns the relative number of functional requirements that are evaluated in two successive feedback events. That is, it indicates whether the designer is evaluating an increasing, decreasing, or constant number of functional requirements. The identity of the specific FRs is disregarded; both events in the pair could concern the same set of FRs or an entirely different set. However, an expanding number of FRs guarantees that new FRs have found consideration, and the focus on functionality is widening; a decreasing number guarantees that the focus is narrowing.

Episode pattern 6b (Table 4-4) represents an identity pattern similar to parameter identity, but with respect to FRs instead of DPs. Similarly it indicates whether the designer is evaluating the same set of FRs repeatedly or is switching to a different set. This pattern might be referred to as *FR identity*. The total number of FRs is disregarded; each set could consist of a similar number or a different number.

4.3.4 Patterns as a Basis for Measures

The foregoing discussion has examined patterns that emerge from various sequences of three episodes of assimilation or feedback. Despite this limited scope, several patterns were identified that have an interpretation suggestive of an influence on design outcome. These patterns represent a potential focus for those concerned with measuring iterative character in a way that has potential relatability to design outcome. Measures based on analysis of these patterns would allow a researcher to objectively measure the aspect of iterative character that they express, and draw distinctions among observed processes in these terms.

The next section proceeds by developing several measures of iterative character that are based upon these patterns.

4.4 Measures of Iterative Character

4.4.1 Feedback Quality Measure

The parameter quantity and parameter identity measures both relate to an issue that will be called *feedback quality*.

Feedback information represents an opportunity for what might be called "iterative learning", the process by which feedback builds knowledge about the effect of each parameter on performance of the design. This interactive process helps guide the process most effectively toward the goal.

One may now imagine a "random iterator" who, by leaping around randomly, tends to generate confounded feedback information about an ever changing set of parameters, and a "learning iterator" who tends to concentrate feedback on one parameter at a time and immediately reapply the information to the same parameter. The random iterator generates relatively poor quality feedback information compared to the learning iterator. A measure that allows us to distinguish between the random iterator and the learning iterator based upon the quality of feedback they generate would be a potentially valuable measure of iterative character. For instance, it would allow the exploration of research questions such as whether the learning iterator tends to have a more successful design outcome.

A basis for comparing processes in terms of their feedback quality may be constructed from the parameter quantity and parameter identity patterns. A grid is defined with parameter quantity on the horizontal and parameter identity on the vertical, as depicted in Figure 4.11. In this space, processes that tend toward concentrated, stationary feedback patterns would reside toward the upper right hand corner of the grid (Quadrant A) indicating that they tend to generate the highest quality feedback information. Processes that tend toward confounded, traveling feedback patterns would reside toward the lower left corner (Quadrant C) indicating that they tend to generate lower quality feedback information. Quadrants B and D represent intermediate feedback

quality. Quadrant B represents a stationary but confounded pattern in which each submission for feedback is concentrated on the same group of parameters, but the feedback is confounded because more than one parameter is involved. Quadrant D represents a concentrated but traveling pattern in which feedback is concentrated on a single but ever changing parameter.

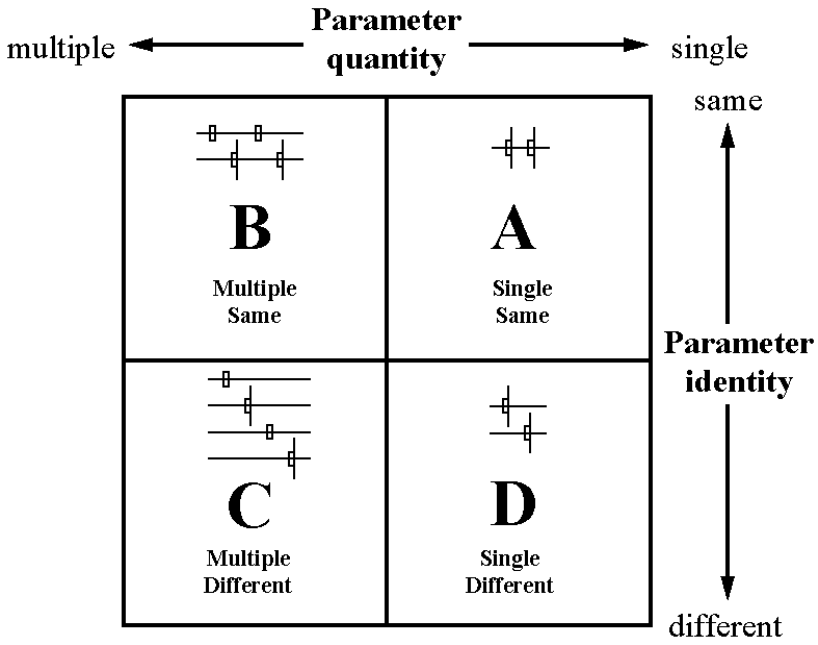


Figure 4.11. Feedback quality measure

The feedback generation pattern of Quadrant A will be referred to a *Type A iteration*. Similarly, the patterns of Quadrants B, C, and D will be referred to as *Type B, Type C, and Type D iteration*, respectively. Several *A-F-DP* timelines depicting examples of processes possessing these feedback patterns are found in Figures 4.12 through 4.15.

Figure 4.12 depicts a process that is predominantly Type A because most successive events involve the same single parameter. It is predominantly but not purely Type A because a minority of Type D events are generated due to the periodic need to transition from one parameter to the next. Although a process cannot be purely Type A if it involves more than one parameter, a pure Type A state can be approached if each parameter is continued a large number of times.

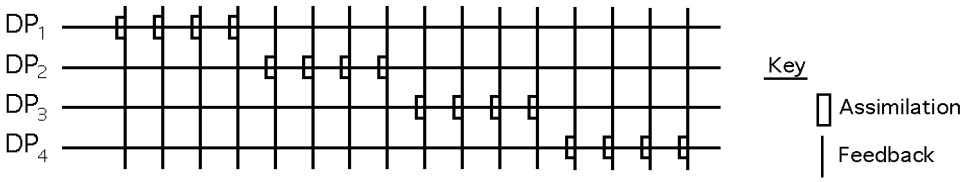


Figure 4.12. Type A (quantity single, identity same)

Processes that are dominant in Type A episodes would naturally be associated with optimization of the effect of a single parameter. This involves the generation of unconfounded information about the effect of a parameter change and immediate application of learning about the effect.

The process of Figure 4.13 is entirely Type D because all of its successive events involve a single but different parameter. A Type D process does not follow an optimizing pattern. It tends to generate unconfounded information about the effect of a single parameter change, but does not tend to reapply the lesson immediately to the same parameter.

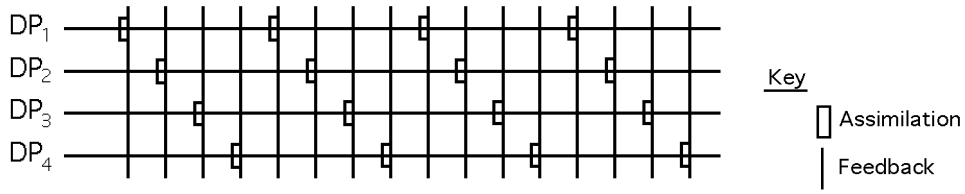


Figure 4.13. Type D (quantity single, identity different)

The process of Figure 4.14 is predominantly Type B because most of its successive events involve the same set of multiple parameters. It is predominantly but not purely Type B because the transition between the DP1:DP2 pairing to the DP3:DP4 pairing is a Type C event.

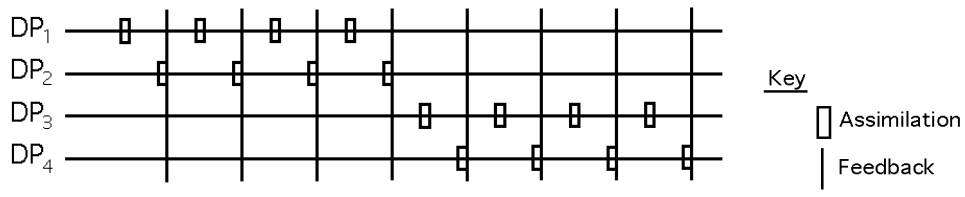


Figure 4.14. Type B (quantity multiple, identity same)

Like Type A processes, processes that are dominant in Type B episodes also follow an optimizing pattern in which the same parameters are present in successive episodes, implying that learning about the effect of the set of parameter changes is being reapplied. However, the feedback this generates is confounded and thus of a different quality.

The process of Figure 4.15 is entirely Type C because its successive events involve different sets of multiple parameters. Type C processes do not follow an optimizing pattern nor do they immediately reapply what is learned.

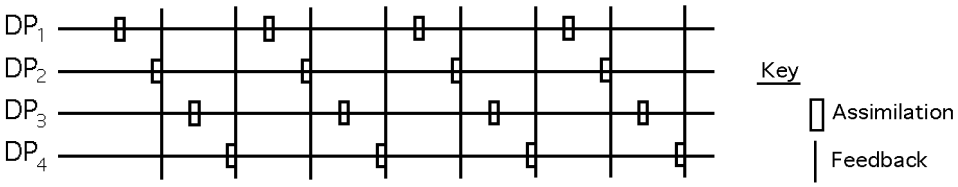


Figure 4.15. Type C (quantity multiple, identity different)

4.4.2 Assessing Feedback Quality

There are at least two ways to assess a design process in terms of Type A, B, C, and D feedback quality patterns.

Discrete Assessment of Feedback Quality

One way is by simply examining each feedback event, determining its classification as Type A, B, C, or D, and computing the percentage of feedback events in each category over the entire process. The categorization may be made objectively according to the following process:

1. Determine the number of parameters edited between the current and previous feedback events.
2. For each parameter, determine whether or not it is also edited prior to the next feedback event (i.e., *continued*).
3. If only one parameter is edited, and it is continued in the next feedback event, the feedback event is Type A.
4. If only one parameter is edited, but it is not continued, the event is Type D.

5. If more than one parameter is edited, assign classes as follows:

   If all parameters are continued, assign Type B.

   If no parameters are continued, assign Type C.

   If some are continued and some are not, the feedback event consists of a Type B component and a Type C component. Classify the event in an apportioned manner as follows: Assign Type B to each parameter that is continued and Type C to each parameter that is not, and divide each by the total number of parameters. For example, if one parameter is continued and two are not, the event consists of 1/3 Type B and 2/3 Type C.

Once every feedback event in the process has been classified, the relative percentages of classes A-D may then be placed in the corresponding quadrants of a 2x2 grid, resulting in a *feedback quality profile*. For example, if the process is 63% A, 4% B, 20% C and 13% D, the grid would be filled as shown in Figure 4.16:

4	63
20	13

%B	%A
%C	%D

Key

Figure 4.16. Example feedback quality profile

Similarly, the example processes of Figures 4.12 through 4.15 would be depicted as shown in Figure 4.17:

0	80
0	20

fig. 4.12

0	0
0	100

fig. 4.13

86	0
14	0

fig. 4.14

0	0
100	0

fig. 4.15

%B	%A
%C	%D

Key

Figure 4.17. Feedback quality profiles for processes of Figures 4.12 through 4.15

Once a feedback quality profile has been created for each process in a group of observed processes, the profiles may then be grouped based on their similarity in terms of the four quadrant values. Several approaches for this task will be described later.

Computed Assessment of Feedback Quality

One potential criticism of the discrete method concerns its tendency to weight all confounded feedback events equally, regardless of the number of parameters that actually confound the event. One may instead wish to account for the difference between events confounded by, say, two parameters versus three or more, on the theory that a small number of confounding parameters delivers more useful information than a large number. A computed approach to assessing feedback quality of a process would characterize parameter quantity by computing an average value for parameter quantity over the whole process, rather than simply classifying feedback events as involving one or multiple parameters.

Initial trials with this approach suggest that its result is not dramatically different from that of the discrete approach. Furthermore, the computed nature of the result prevents it from being expressed in the quadrant-based classification system. A recommended procedure for conducting the computed approach, as well as the results of the trial application, are described in Appendix E.

4.5 Drawing Comparisons Among Processes

Applying the feedback quality measure is an important step toward drawing distinctions among individual processes. However, application of a measure to individual processes allows little more than pairwise comparison between two individuals; comparing multiple individuals simultaneously remains difficult. An effective way to draw comparisons is to *group* similar individuals into one of a set of categories based on their *similarity* in terms of the measure. Then, comparisons may be drawn among each group while retaining the ability to inspect individuals as needed.

The first concern in grouping individual processes is the choice of a basis on which to evaluate their similarity. Similarity may be evaluated with respect to either similarity in pattern or similarity in feedback quality. *Pattern* simply refers to the relative distribution of A, B, C, and D patterns of which a process is composed. A feedback quality profile expresses this pattern distribution directly. Alternatively, *feedback quality* is an overall

quality assigned to the process based on its distribution of A, B, C, and D patterns and the theoretical relative quality contribution of each. Type A patterns contribute the highest quality feedback because they maximize parameter continuity and minimize parameter confounding. Type B and D patterns contribute intermediate quality feedback because each possesses one of the effective aspects of the Type A pattern but lacks the other. Type C patterns possess neither and so contribute the lowest quality feedback. A given process might be assigned an overall feedback quality by computing a simple weighted sum based on its pattern distribution and these relative quality levels.

The next concern is that of grouping processes with respect to their similarity. The general problem of grouping items based on similarity is a problem of classification. If an existing classification scheme is known to apply to the items that are to be grouped, classification can be done in a straightforward manner by a distance-based, least squares approach. For example, if the items to be classified are samples of unidentified construction material, then the applicable categories would correspond to the set of known types of construction materials. If each sample is described in terms of variables representing its physical properties, it may be assigned to a category by computing a Euclidean distance between these variables and the geometric centroid of each respective category. The item may then be assigned to the geometrically nearest category.

In classification problems where an applicable classification is not known in advance, a number of approaches are available. One possibility is to simply specify *a priori* a set of categories based on canonical theory, intuition, or an examination of the data. Items may then be assigned to the geometrically nearest category. If no canonical theory or other basis is available, analytical techniques may be employed to divide the items into so-called "natural" categories that are implied by the distribution of the items themselves. This is known as *cluster analysis*, and is described in detail where it is employed in Appendix E.

The current study assigns individual processes to a set of canonical categories that are suggested by a simple naming convention. Processes may be assigned to these categories either by manual application of the naming convention or by a distance-based method. The following sections describe these methods in more detail.

4.5.1 Grouping by Similarity in Pattern

The X+yz Naming Convention

As a first-order method to categorize processes manually, a naming convention was developed. The naming convention simply characterizes each process in terms of its most dominant and next most dominant quadrants. The notation takes the form

$$X + yz$$

in which *X* is an upper-case letter representing the strongest quadrant, and *yz* are lower case letters representing secondary quadrants.

The guiding principle in this notation is to specify the letters of each quadrant, starting with the most dominant, until 75% of the patterns in the process have been accounted for. The letter of the strongest quadrant is placed in the first position, as a capital letter to indicate that it is dominant. If this quadrant is 75% or greater, no more letters need be assigned. Otherwise, the second largest quadrant is indicated by a lower case letter. If the sum of the percentages in the first and second quadrant add up to 75% or more, no more letters are added. Otherwise the third largest quadrant is added in lower case, and the remaining quadrant is disregarded. Figure 4.18 shows an example data set of four hypothetical processes to which this naming convention has been applied.

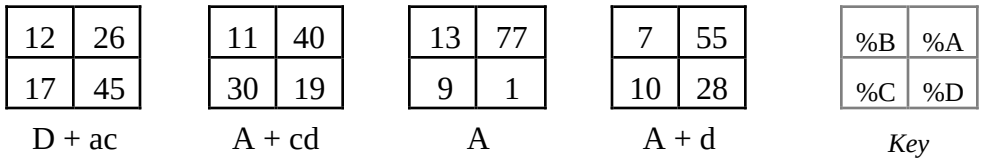


Figure 4.18. Example feedback quality profiles with *X+yz* naming

Ties are governed by the following rules. In the case of a tie in selecting the first letter, the next largest quadrant breaks the tie in favor of the quadrant to which its pattern is most similar: for example, if A and D are tied, then C favors D and results in the choice of D+a. In an A-C or B-D tie, this rule is inconclusive and in this case A favors C and D favors B by convention. In the case of a tie selecting the second letter, the quadrant that

is by definition higher in feedback quality is selected (choosing A over B/C/D, and choosing B/D over C). If the tie is between B and D, which are both intermediate quality, D is selected if A is larger than C, and B is selected otherwise. In the case of a tie in assigning a third letter, neither letter is assigned because mathematically the third number can be no larger than 16%, and is either much smaller or is relatively balanced with the fourth number, which limits its discriminative power. Finally, if more than one quadrant is capable of reaching the 75% threshhold and thus halting the naming process, the quadrant with the larger percentage is selected.

The threshholds employed were carefully selected to provide certain interpretive properties to the naming convention. The presence of a single capital letter not only indicates that a process is particularly strong in that quadrant (at least 75%), it also implies that no other single quadrant can possess more than the remaining 25%, and likely contains less if it is shared with other quadrants. The addition of a single lower case letter indicates that the dominant quadrant is less than 75%, and the secondary quadrant is quite strong because only two letters needed to be assigned to fulfill the 75% threshhold. Because the first quadrant must be larger than the second, the dominant quadrant is no smaller than 38% and will usually be greater. The presence of a third letter implies that three quadrants were sufficiently similar in magnitude that all of them were required to reach the 75% threshhold, and by implication the dominant quadrant is not particularly dominant.

The *X+yz* naming convention is intended to provide a simple means of manual classification that allows a first-order grouping while preserving specific information about each individual. For example, individuals might be grouped according to their dominant element(s) *X* or *X + y*, while remaining distinguishable within each group by reference to their less significant elements *y* and/or *z*.

Canonical Profile Categories

Profiled processes might alternatively be grouped analytically by a least-squares, distance-based method. This requires that a set of predetermined categories be

established and assigned geometric centroids to that distances from each item to each category may be computed.

As suggested above, individual processes may be grouped according to their dominant element(s) *X* or *X + y* while reserving *z* for the purpose of within-group distinction. This approach was adopted for the current study. A total of sixteen canonical categories result from this *X+y* categorization.

Geometric centroids were assigned to each canonical category based upon the averages of the ranges of quadrant values they represent. Each of the four single-letter categories (i.e., *X* only) were assigned a centroid defined by 88% in the dominant quadrant, with the other three quadrants evenly dividing the remainder. The 88% figure represents the midpoint of the 75%-100% range that defines a single-letter category. Similarly, two-letter processes were assigned a centroid defined by 63% in the dominant quadrant and 19% in the next dominant quadrant, with the other quadrants dividing the remainder. The 63% figure represents the midpoint of the 51%-75% range that would characterize the dominant quadrant in this case. The 19% figure represents the midpoint of the remaining 37% that is available for the secondary quadrant. Table 4-5 summarizes the resulting set of centroids for each canonical category.

Table 4-5. Centroids of canonical categories based on an *X+y* naming convention

	%A	%B	%C	%D
A	88	4	4	4
A+b	63	19	9	9
A+c	63	9	19	9
A+d	63	9	9	19
B	4	88	4	4
B+a	19	63	9	9
B+c	9	63	19	9
B+d	9	63	9	19

	%A	%B	%C	%D
C	4	4	88	4
C+a	19	9	63	9
C+b	9	19	63	9
C+d	9	9	63	19
D	4	4	4	88
D+a	19	9	9	63
D+b	9	19	9	63
D+c	9	9	19	63

4.5.2 Ordering Groups by Similarity in Feedback Quality

The preceding discussion has concerned the grouping of subjects based on similarities in their pattern distributions. Because pattern distribution is directly related to feedback quality, a grouping based on pattern distribution can be made to express similarity in feedback quality as well, by ordering the categories according to their relative feedback quality. Having defined the centroids of each canonical category, each may be evaluated and ranked so that categories having a similar quality may be placed near each other in a classification chart.

Each canonical category was assigned a relative feedback quality rank by computing a weighted sum of its defined pattern distribution. Weights were selected to reflect the general quality level associated with each pattern. Type A was given a weight of 2 to reflect the two aspects that account for its high feedback quality (minimum parameter quantity and maximum parameter identity). Types B and D were both assigned a weight of 1 to reflect the presence of only one of these aspects. Type C was given a weight of zero because it lacks either aspect. This resulted in the following scoring formula:

$$\text{feedback quality score} = 2(A) + 1*(B+D).$$

This formula was then used to compute a feedback quality score for each canonical category, with A, B, and D representing the defined centroid values for each category. For instance, the category A+d has centroid values of A = 63, B = 9, and D = 19, leading to a relative feedback quality rank of 154. Applying the above scoring formula to all sixteen canonical categories results in nine quality ranks as shown in Table 4-6, ranging from the highest feedback quality (A) to lowest (C).

Table 4-6. Quality ranking of canonical categories

1	2	3	4	5	6	7	8	9
A	A+d A+b	A+c	D+a B+a	D B D+b B+d	D+c B+c	C+a	C+d C+b	C

A classification chart may then be constructed to reflect this quality ranking, as shown in Figure 4.19. When individual processes are placed within this chart (as indicated by the symbol ###), they may then be distinguished not only by their placement within a certain canonical category (which communicates similarity in pattern distribution) but also by the position of their category within the chart (which communicates similarity in feedback quality).

Pattern:	**A**	**A+d**	**A+b**	**A+c**	**D+a**	**B+a**	**...**
	###	###	###	###	###	###	
		###	###	###			
		###					...
		###					
		###					
Quality Rank:	**1**	**2**		**3**	**4**		**...**

Figure 4.19. Portion of the feedback quality classification format

4.6 Conclusion

The discrete feedback quality measure was developed as a measure of iterative character that is potentially relevant to issues concerning design outcome. Application of the measure to an individual design process involves the construction of a design timeline, the analysis of the timeline in terms of parameter quantity and parameter identity patterns, and the conversion of these patterns into a 2x2 feedback quality profile. Profiles of individual processes may then be grouped based on similarities in pattern distribution and in feedback quality by applying the *X+yz* naming convention and placing named processes into the classification chart of Figure 4.19.

In closing this chapter, we now turn toward the issue of applying the measure to empirical data. The remaining chapters of this document describe the empirical phase of this study, in which data is gathered and measured in a test application of the feedback quality measure. In Chapter 5, a data collection instrument is developed for the purpose of gathering empirical data. The development of this instrument also serves as an

example of how a parametric design task may be instrumented. Chapter 6 describes the use of this instrument in a data gathering exercise, which results in the collection of two distinct sets of data. Chapter 7 describes the application of the measure to this data and builds a validity argument based on the results of the analysis.

5 A Data Collection Instrument

Having proposed a measure of iterative character, the picture is not complete without addressing its practical application to research. This chapter has two goals. First, it describes a data collection instrument that was used in the current study to gather data with which the feedback quality measure of iterative character could be validated. The goal here is simply to make it possible for others to understand the origin of this data. Second and more importantly, it addresses the conditions for application of the measure in other experimental settings, so that other instruments may be developed around other experimental design tasks, using this instrument as an example.

By necessity, the development of the data collection instrument described in this chapter took place early in the project so that empirical data could be gathered to help inform the development and selection of the measure. The considerations that guided the choice and development of this instrument apply independently of the final form of the measure and are instructive toward the development of different instruments involving different experimental design tasks. The discussion in this chapter reconstructs these considerations and presents them as if the instrument is being constructed for the first time. This perspective allows the discussion to focus on the considerations that apply to the selection and development of any similar instrument and to provide a complete example of the steps involved in doing so.

5.1 Selecting a Subject Design Task

The first question in constructing an instrument is that of selecting an experimental design task that the subjects are to perform and from which the instrument will record data. In the current study, some of the considerations in this choice relate to the goal of validating the measure. These include the choice of a particularly iterative design task so that the data collected will be likely to have a strong iterative character that can easily be detected. Another consideration was to select a task that could be easily instrumented to collect data automatically rather than through a more laborious method such as verbal protocol analysis. Therefore the process about to be described applies particularly to

parametrically structured design tasks that have been implemented as interactive software.

These considerations acted in addition to more general concerns relating to the type of data that must be collected to apply the measure in a more generic experimental setting. Development of a new instrument would more likely be dominated by these concerns. First and most importantly, the design task should provide opportunity for variations in its iterative solution. Second, the design task should be one in which assimilation and feedback events may easily be recognized and associated with specific design parameters.

These preferences suggest a *virtual prototyping* design task. Virtual prototyping refers to the representation of a design artifact as a geometric computer model in a sufficiently complete form that issues such as interference, physical properties, and performance of the artifact can be predicted via the model without the need for construction and evaluation of a physical prototype [Calkins et al. 1998]. This approach is increasingly popular among engineering design firms [Adam 1993], [Blanchard 1996], [Beckert 1996].

To the designer, one of the primary advantages of virtual prototyping is the ability to generate and evaluate alternative designs without building a physical prototype of each. Whereas the traditional building and testing of physical prototypes would involve a significant amount of physical manipulation and social interaction that would have to be manually observed and interpreted, the emphasis on interactive computer modeling means that a large portion of the designer's activity leaves evidence in the form of keystrokes, mouse clicks, screen views, and the like. Furthermore, because both generation and evaluation take place within this environment, feedback and its effect on subsequent design decisions might specifically be traced.

Unfortunately, virtual prototyping systems that are in use in professional settings [King 1998, Stewart and Hallenbeck 1997] are proprietary, and so are not readily accessible for instrumentation for the purpose of data collection. Also, these systems are quite complex and require significant training to become proficient, which would tend to limit the population of potential subjects.

An alternative to a professional virtual prototyping system might be found in a parametrically structured design task that has been computationally modeled. Computationally modeling a parametric design problem allows one to generate and evaluate candidate solutions by entering various parameter values and computing resultant performance on demand. This leads to the availability of feedback at a very low cost, and suggests that a computer-modeled parametric design problem would support if not encourage a particularly iterative, feedback-reliant design process. Parametrically structured design problems have a high degree of structure that leads to a relatively well-bounded solution process. This also makes parametric problems good candidates for modeling computationally because the performance of a proposed design may generally be predicted as a function of its parameter values.

A number of realistically complex design problems have been modeled parametrically and then computationally modeled in an interactive computer-based design environment. One example is West Point Bridge Designer [NEEDS 2000], an educational software application that allows students to specify and test designs for a truss bridge. This problem is presented parametrically in that the component materials, their dimensions and other properties, and their physical relationships may be specified by the designer, resulting in a candidate solution that may be tested and modified at will. One of the stated advantages of the software is the enabling of more iterations than would be possible if the building of a physical prototype were the only means of evaluating candidate designs [Ressler 2000].

Parametric design tools can also be found in recreational design applications that are increasingly diverse. For example, one web-based tool allows consumers to design custom shoes interactively by selecting various zones on a generic shoe design and assigning any of dozens of possible design options to each, resulting in a dynamically generated rendering of the resultant design and its purchase price [Goldberg 2000]. Another allows home brewers to design recipes for beer by entering ingredient quantities and other variables that determine color, alcohol content, and style characteristics [Riley 1998]. These examples suggest that a large population of computationally modeled parametric design tasks exist that could potentially be instrumented to serve as platforms for the gathering of empirical data.

5.2 *Virtual Car* Educational Software

University courses in engineering design can provide a rich setting for empirical research on the design process. At the University of Washington, one example of such a course is Engineering 100, Introduction to Engineering Design. ENGR 100 is an elective course open to freshman engineering students and others interested in engineering design [Safoutin et al. 2000], [Kramlich and Fridley 1998]. The course is primarily activity-based, with an emphasis on two group design projects of four to five weeks duration that include construction and testing of student designs in a competitive atmosphere.

One of the projects taught in this course employs an educational design software application that follows a parametric design model. As an instructor of this course, the author of the current study developed this application to demonstrate the virtual prototyping approach to engineering design. The software, called *Virtual Car*, allows students to design a small toy car in a parametric manner, evaluate candidate designs in terms of projected performance, and create construction templates that can be used to build a faithful and fully operative physical version of a modeled design. The software has been used for several years in this course, and has become available to other colleges and universities nationwide as a result of its selection as a finalist candidate in an educational software competition [NEEDS 2001].

A large proportion of the Virtual Car design process is performed in direct interaction with the software. Because of this, a large proportion of the designer's activity is likely to take place in the form of user interaction with the software and so might be automatically captured by instrumenting the software to record such activity. This suggests that the iterative component of the design activity may also be captured if the software were properly instrumented.

5.2.1 The Design Task

The task presented to a designer using Virtual Car is to design a small toy car that is powered by a wind-up spring (also known as a power spring or clock spring). Figure 5.1

illustrates the components of a typical Virtual Car. The body of the car is built from layers of a thick sheet material such as foamcore or corrugated cardboard. Two distinct body layer shapes are employed and can be seen in the figure. Two layers of an "outer" shape form the two side walls of the vehicle and provide bearings for the front and rear axles. A variable number of layers of an "inner" shape form the inner core of the car. These layers define the form of the longitudinal center of the vehicle body, and include a circular or octagonal cutout which form a chamber for the spring.

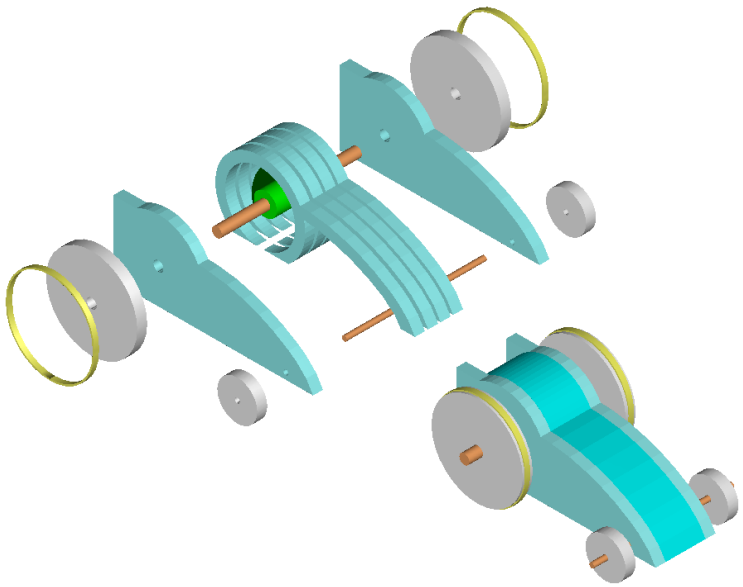


Figure 5.1. Functional components of a Virtual Car

The spring engine, a wound-up strip of springy plastic or metal, resides in the chamber formed by the inner and outer layers. One end of the spring engine is connected to the drive axle and the other is anchored to the inner surface of the spring chamber. The spring may be any width or length specified by the designer, and typically is fabricated from plastic material cut from a plastic soda bottle. In the figure, a rear wheel drive configuration is depicted, the spring being wound to propel the car to the right. A front-wheel drive configuration would wind the spring in the opposite direction to travel leftward. In all-wheel drive versions, an additional spring chamber exists at the opposite end of the vehicle, identical to the first chamber. The shape of the inner and outer layers is specified directly by the designer, with the exception of the spring chamber which is superimposed upon and supercedes the specified shape. The size and location of the

spring chamber is computed by the software as a function of specified spring length, drive configuration, and other parameters, and then superimposed on the specified inner shape.

Typical design goals are either to maximize the speed with which the car will reach a finish line 15 feet away, or to maximize the total travel distance before the spring becomes unwound. A valid design must avoid skidding of the drive wheels due to too little traction or too much power. Constraints include a limit on car size and wheel size (to accomodate the printing of construction templates on a standard sheet of paper), and a limitation on the width of the spring (six inches).

The Virtual Car design software provides a simple design environment with which to conduct the design process. It presents the designer with a fully parameterized representation of the design task, allowing the designer to specify parameter values interactively and request immediate feedback about various aspects of the resultant performance of the evolving design. Despite a relatively small number of design parameters, a very large number of alternative designs are possible, generating many opportunities to carry out iterative redesign cycles.

The designer is initially provided with a set of default designs that have moderate performance. The designer proceeds by modifying one or more of the default designs to produce a more competitive design. This typically involves iterating through many candidate designs by experimenting with different parameter values, such as car shape, wheel size, spring dimensions, and type of drive (front, rear, or all-wheel). Each combination of parameter values represents a design alternative for which feedback can be gained instantly about performance metrics such as appearance, speed, and travel distance.

Once an acceptable design has been found, a faithful physical prototype may be constructed by printing parts-cutting templates on an ordinary printer and cutting them out of a stackable material to form the body, in a layered method similar to 3-D printing or rapid prototyping [Thilmany 2001]. Figures 5.2 and 5.3 illustrate these steps and show a fully assembled example prototype.

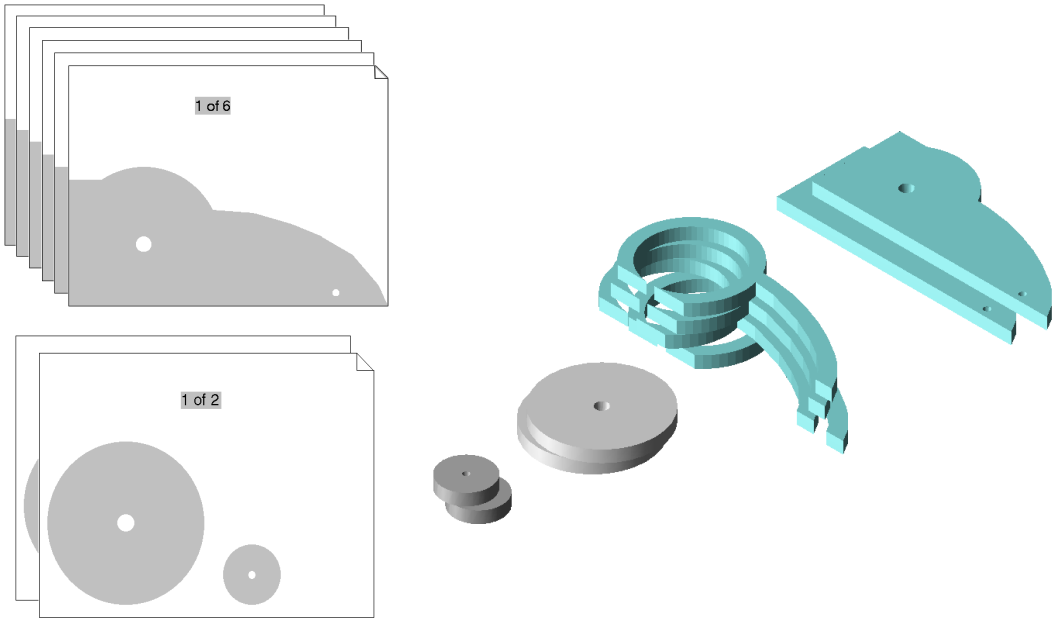


Figure 5.2. Paper parts-cutting templates and finished parts

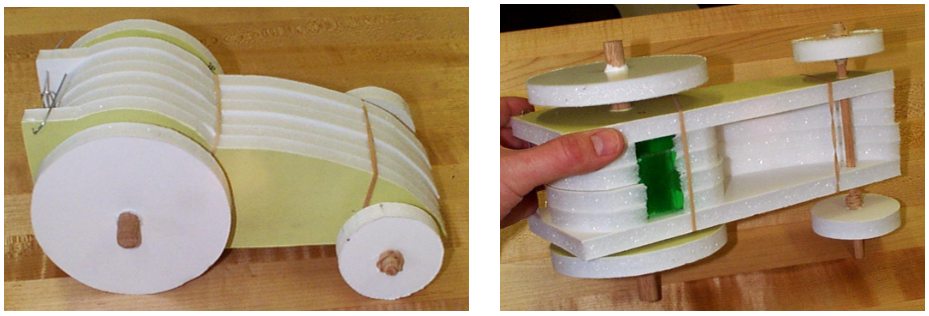


Figure 5.3. Fully assembled Virtual Car

The interface centers around a main design interface by which parameter values are specified, and several feedback modes that deliver various sorts of information about the performance of the design that is described by the current parameter settings. The interface is depicted in Figures 5.4 through 5.9.

The Design mode is depicted in Figure 5.4. It provides a parameter selection interface that allows design parameter values to be specified and thus assimilated into the design. On selection of a design parameter button, appropriate controls such as edit

fields, drawing tools, or buttons appear, with which the designer may specify a parameter value.

Figure 5.5 depicts the first feedback mode, the Build mode. It provides feedback on visual appearance and relationship of components only.

The Analyze mode is depicted in Figure 5.6. It provides feedback on mass, center of mass, and normal forces on the wheel-ground interfaces. These are useful for anticipating vehicle stability and traction, although they do not report actual performance.

The Simulate mode, depicted in Figure 5.7, provides performance feedback. It displays computed performance metrics including time to finish line, maximum powered distance, and top speed. It also provides the opportunity to step through an animation showing variations in traction and propulsion during powered travel.

The Race mode shown in Figure 5.8 provides feedback on speed relative to other modeled designs. It displays a set of animated bars representing the movement of all currently defined vehicles, as if competing in a race. Travel distance is also reported, but only if the vehicle has insufficient distance capacity to reach the finish line (15 feet).

Figure 5.9 shows the Preview mode, which allows the designer to confirm the number and printability of the body construction templates.

Figure 5.10 depicts the Print interface, which does not directly provide feedback, but simply allows the designer to print a report that summarizes the design, or to print its construction templates. Use of the Print interface suggests a decision to construct a physical prototype of the design in order to gain real-world performance feedback. Figure 5.11 shows a printed report for the "Racer" default design, describing its parameter settings and performance characteristics. Figure 5.12 shows a set of printed templates for this design.

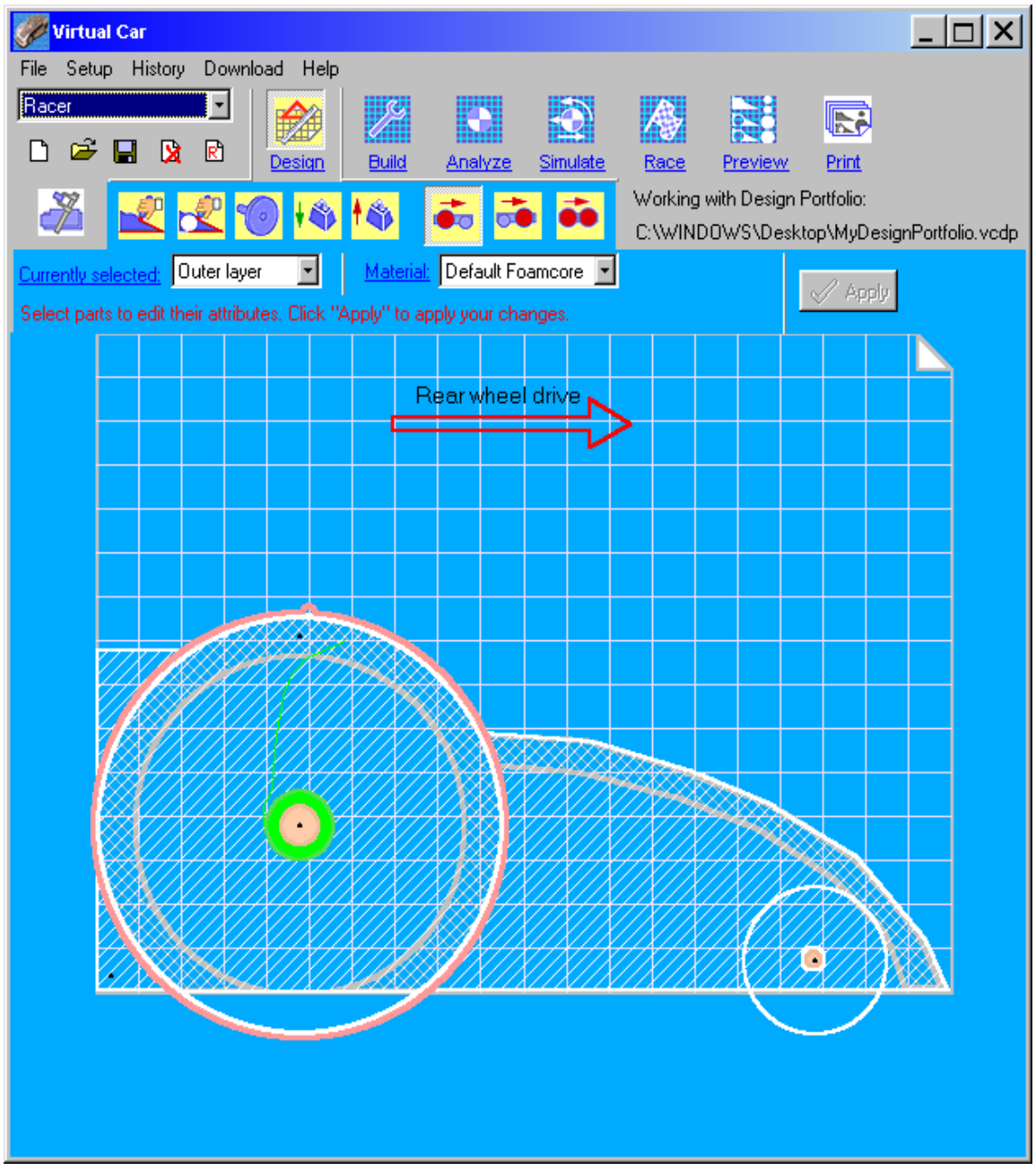


Figure 5.4. Design mode

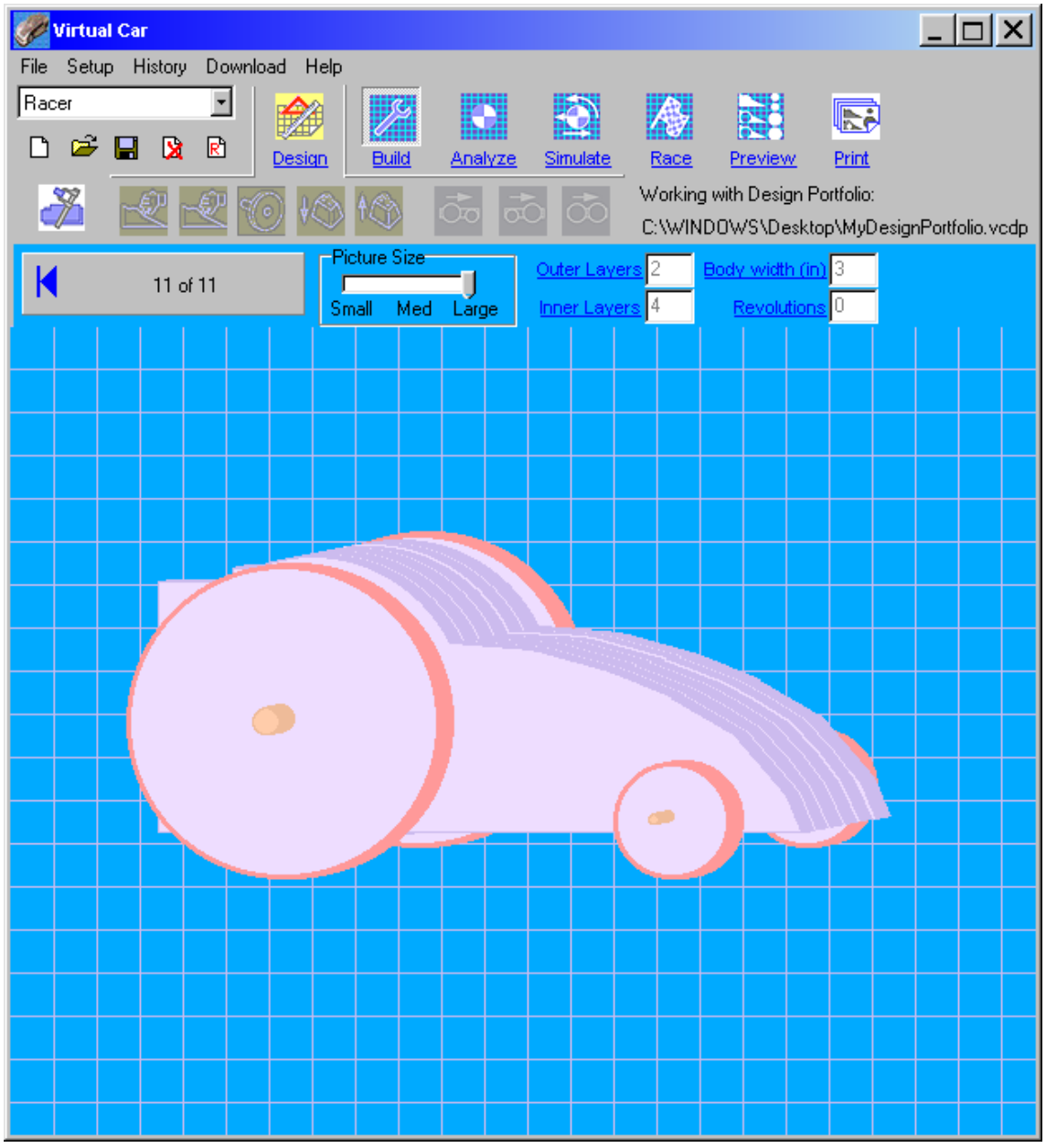


Figure 5.5. Build mode

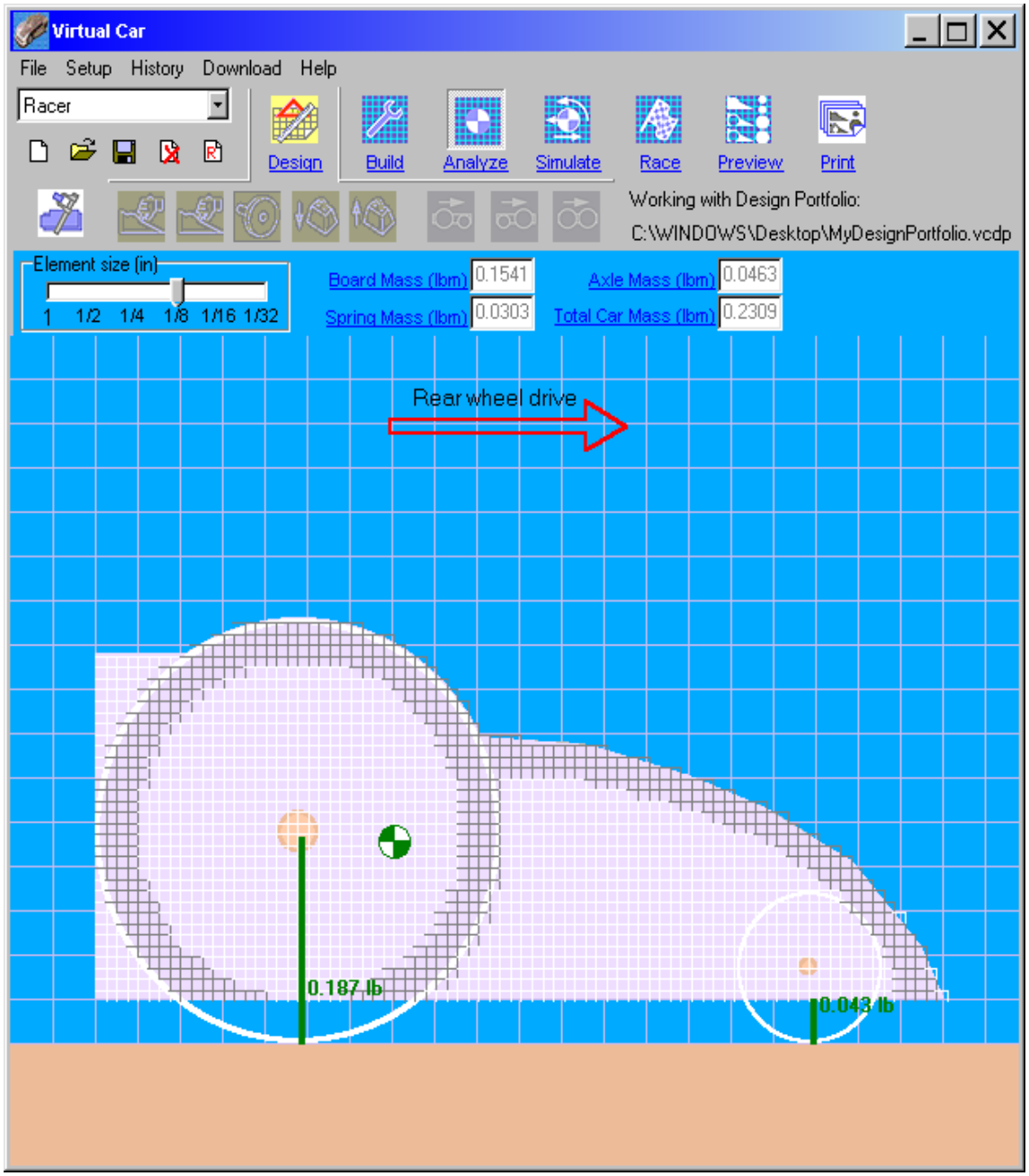


Figure 5.6. Analyze mode

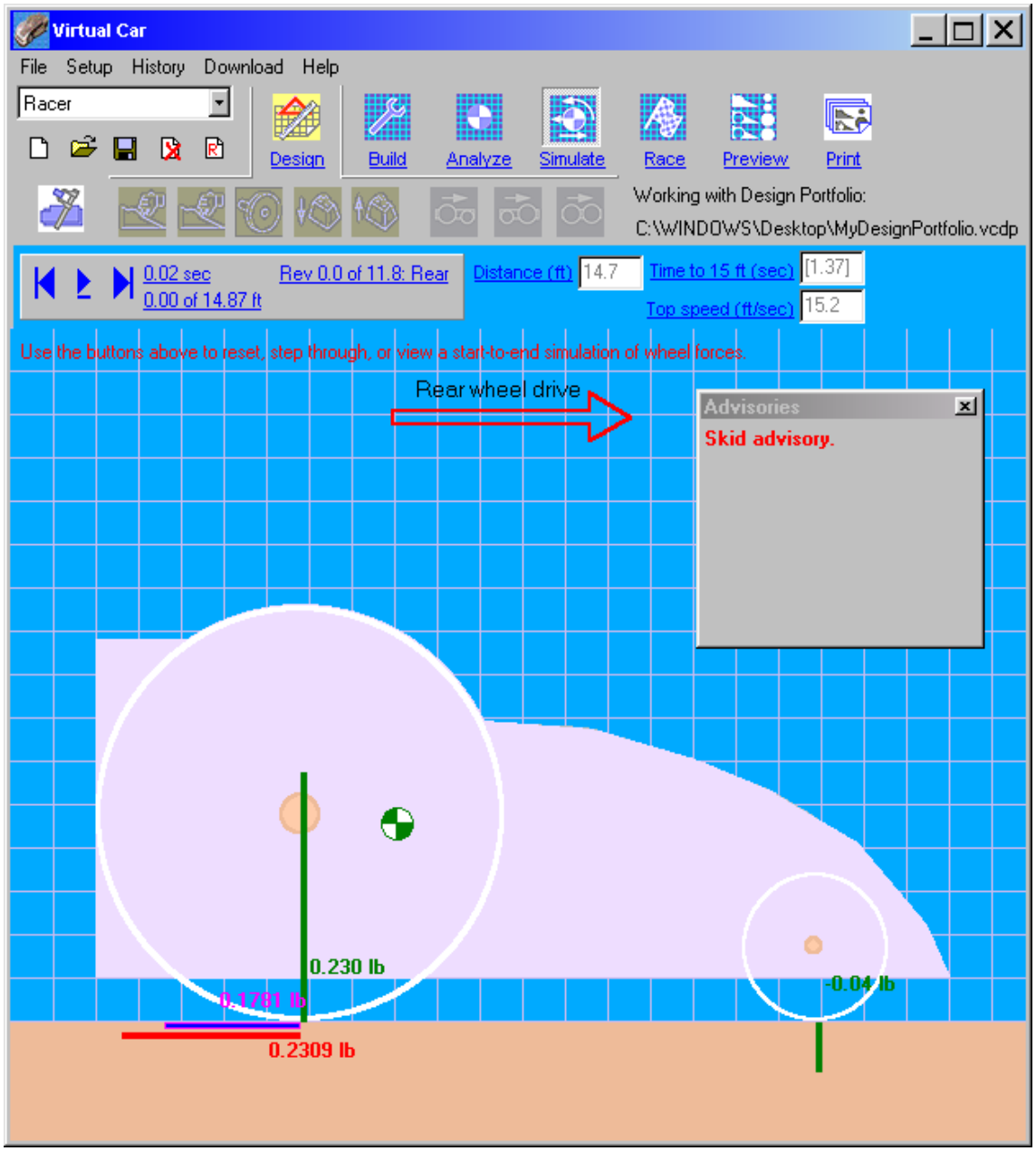


Figure 5.7. Simulate mode

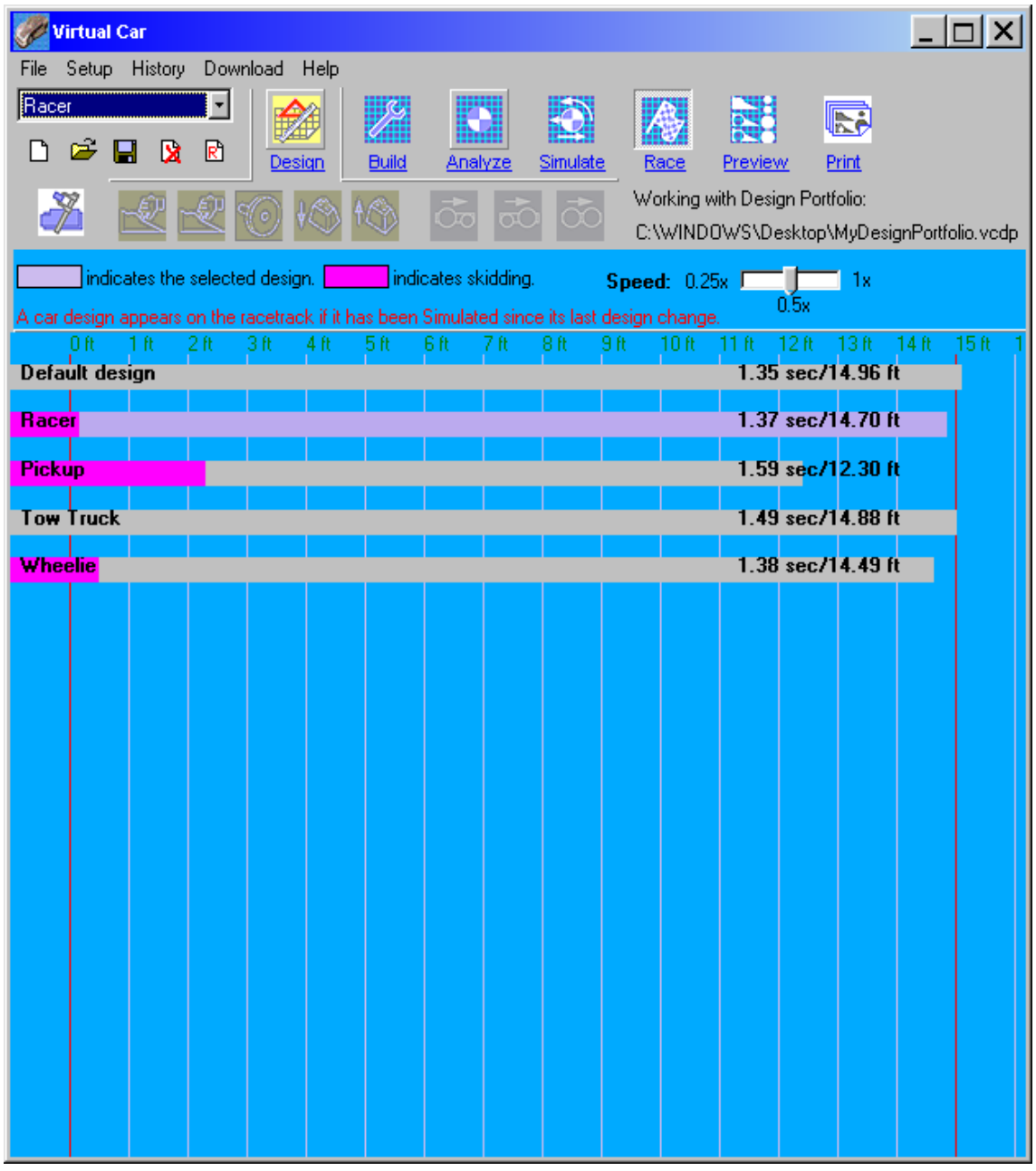


Figure 5.8. Race mode

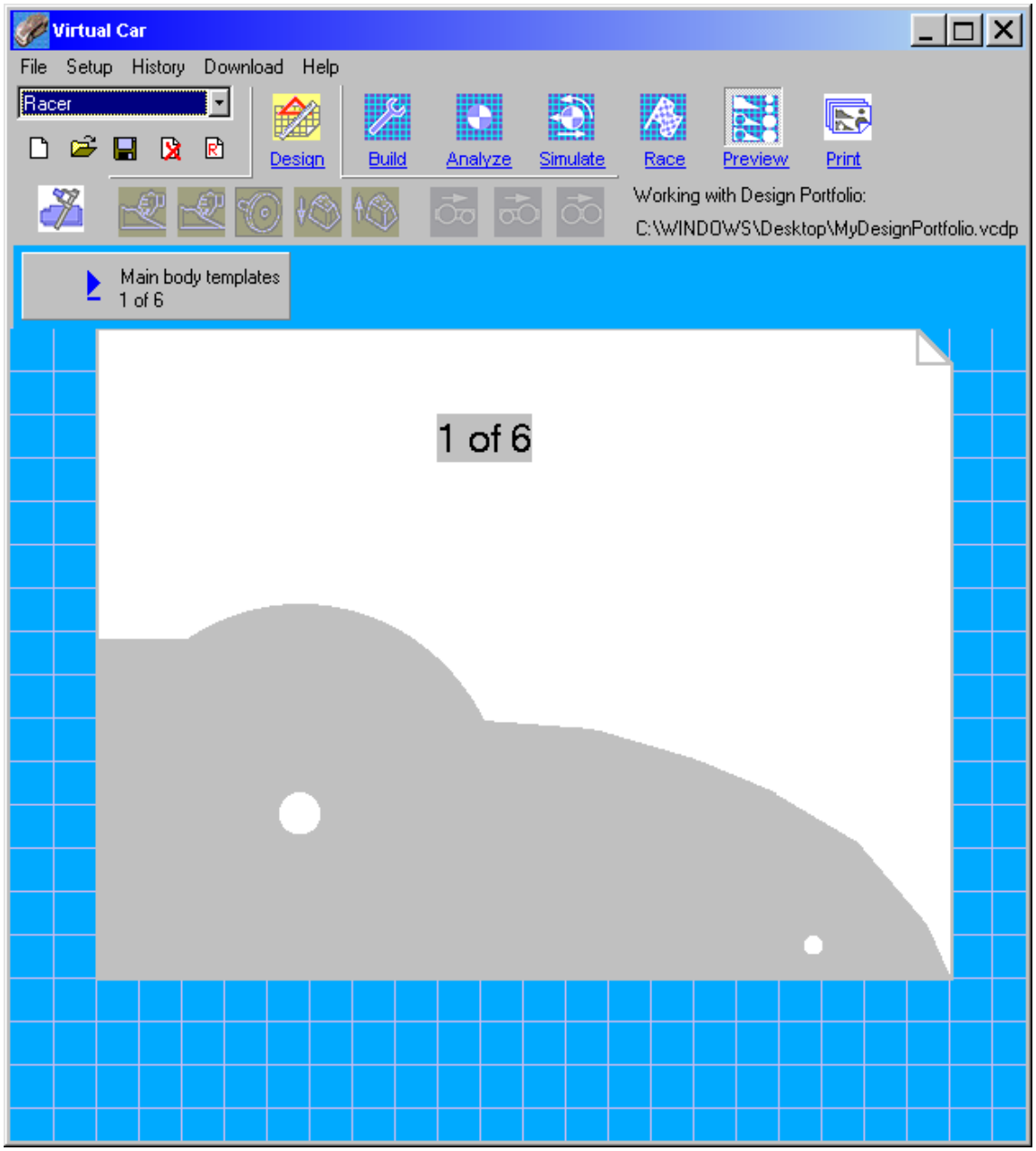


Figure 5.9. Preview mode

Figure 5.10. Print interface

VIRTUAL CAR Design Report for Racer

Design Parameters:

Spring Width: 1.5 in
Spring Length: 36 in
Axle Length: 0 in
Left (Drive) Wheel Dia: 4.80 in
Right (Other) Wheel Dia: 1.75 in
Left (Drive) Axle Dia: 0.5 in
Right (Other) Axle Dia: 0.25 in

Materials and Assumptions:

Outer layers: Default Foamcore (t= 0.50 in, dens= 0.0023 lb/cu in)
Inner layers: Default Foamcore (t= 0.50 in, dens= 0.0023 lb/cu in)
Left (Drive) wheels: Default Foamcore (t= 0.50 in, dens= 0.0023 lb/cu in)
Right (Other) wheels: Default Foamcore (t= 0.50 in, dens=0.0023 lb/cu in)
Left (Drive) axle: Default Oak (0.027 lbm/cu in)
Right (Other) axle: Default Oak (0.027 lbm/cu in)
Spring: Default plastic (t=0.0125 in)
Tread: Ideal Rubber (mu=1)
Maximum torque: 0.57 in-lb (width= 1.5 in)
Finite element size: 9 pixels

Predicted Performance and Physical Characteristics:

Revolutions: 11.84 turns
Predicted propulsive force: 0.24 lb
Predicted traction, left: 0.23 lb
Predicted traction, right: 0.00 lb
Board Mass: 0.15 lbm
Spring Mass: 0.03 lbm
Axle Mass: 0.05 lbm
Total Car Mass: 0.23 lbm

Time to 15 ft:[1.37] sec
Speed at 15 ft:15.20 sec
Projected Distance: 14.70 ft

ADVISORIES:
No advisories

Details:

Predicted Max Torque: 0.57 in-lb
Chamber inner diameter: 3.82 in
Chamber outer diameter: 4.82 in
Chamber width: 2.00 in
Car width: 3.00 in
Total number of layers: 6.00 layers
Cored layers: 4.00
Non-cored layers: 2.00
Bottles needed: 0.75
Overall x-centroid: 3.52 in

Normal force on drive wheel, unwound: 0.19 lb
Normal force on drive wheel, wound: 0.23 lb
Normal force, other wheel: -0.05 lb

Figure 5.11. Printed report

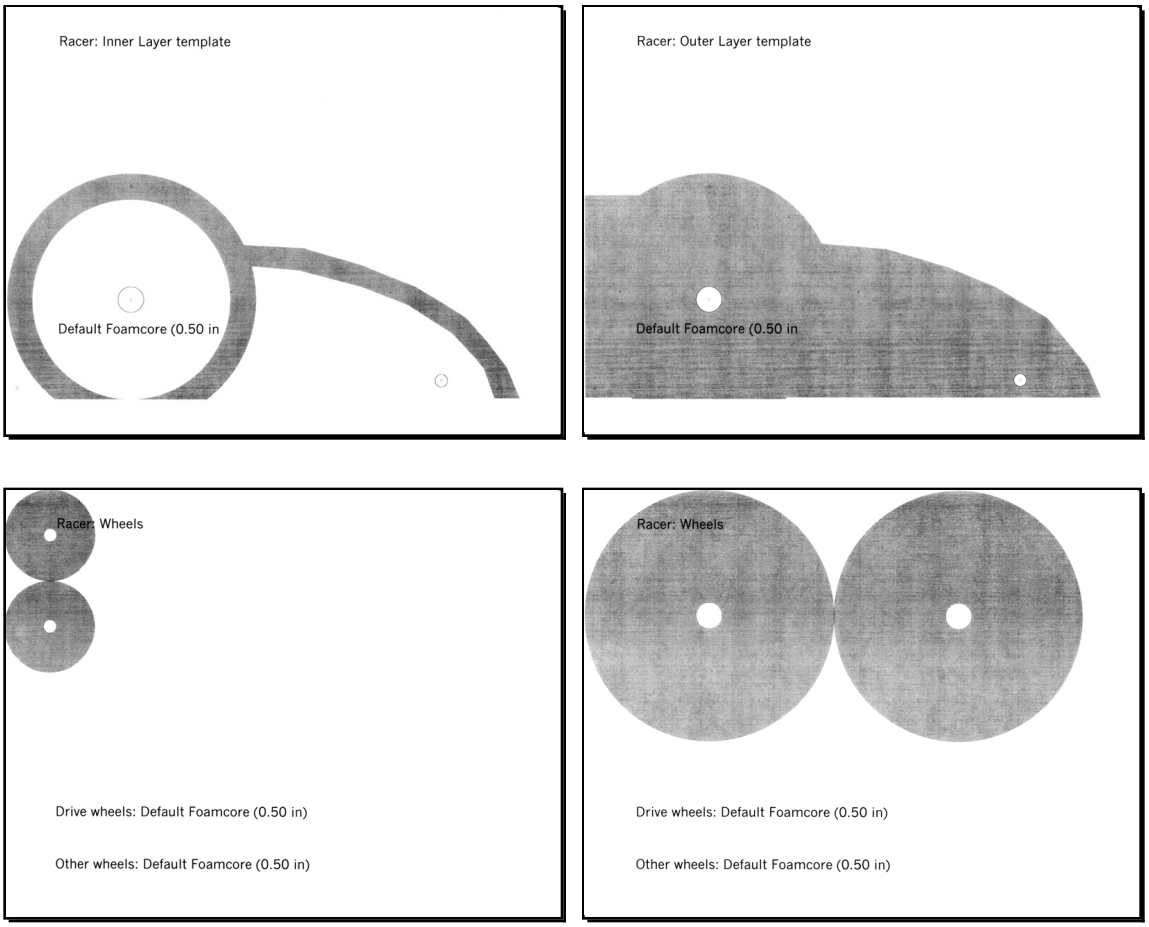


Figure 5.12. Printed templates for Racer design

Virtual Car as a Data Collection Instrument

Virtual Car represents an authentic design problem with a nontrivial solution process. Additionally, its short learning curve makes it an attractive candidate as a platform for collecting empirical data on the design process, particularly in settings that are likely to employ a subject population of novice designers such as undergraduate students.

The task of transforming the software into a data collection instrument consists of two major steps: identification of the design parameters that govern the problem, and the addition of instrumentation to record assimilation and feedback events with respect to these parameters. The next sections describe these steps.

5.2.2 Identifying Design Parameters

The first step in instrumenting a design problem for data collection is the identification of design parameters. The Virtual Car design problem is modeled as a set of twenty-four distinct design parameters, which are depicted in Figure 5.13. Each physical component of the vehicle may have its physical dimensions specified, as well as the type of material of which it is made. Several other parameters specify shapes of vehicle components such as the outer layers, inner layers, and spring chamber, as well as the number and location of spring chambers to specify a rear-wheel-drive, front-wheel-drive, or four-wheel drive vehicle.

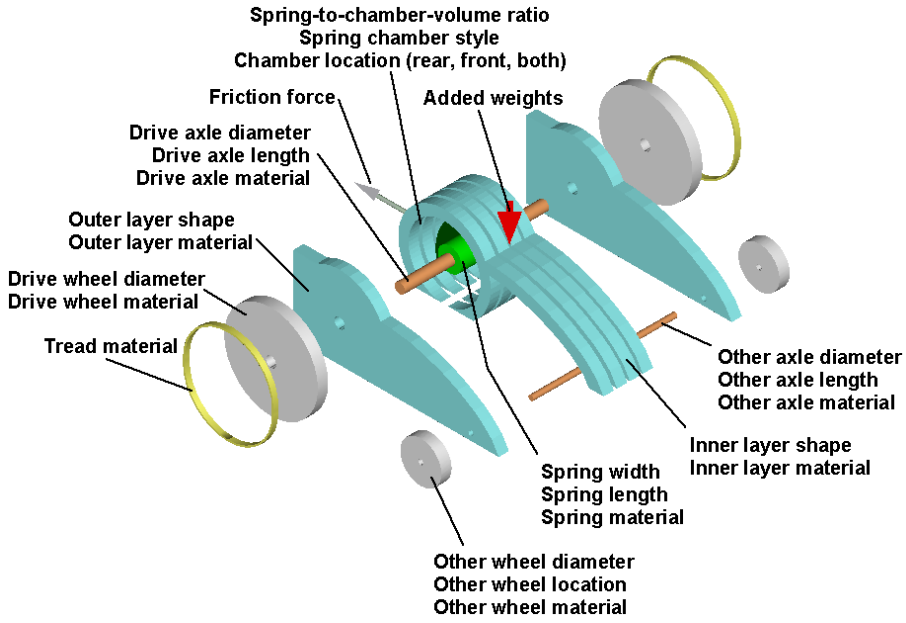


Figure 5.13. Virtual Car design parameters

The design parameters delineate a complete definition of a vehicle design. Many of the parameters may be specified as numeric values, such as spring width or drive wheel diameter. Others such as the shapes that define the car body and the choice of front, rear, or all-wheel drive are non-numeric parameters that are entered by selecting from a list, clicking a radio button, selecting a point on the screen, or drawing a shape interactively with the mouse. Table 5-1 lists the design parameters, their value types, and the means of specification.

5.2.3 Identifying Assimilation and Feedback

The next step is in identifying actions through which assimilation and feedback take place. Because the design artifact is inherently parametric, assimilation is visible in the specification of parameter values. Parameter values are specified in the Design mode via keystrokes, mouse clicks, list selection, or a combination of these actions. Figure 5.14 highlights the interface elements through which assimilation and feedback take place. Assimilation modes are entered via a set of ten buttons and lists, each of which initiate a parameter editing dialog specific to a given component. Feedback modes are entered via six buttons representing: a 3-dimensional rendition for checking of geometry; an analysis of physical properties such as mass, normal force on each wheel, and center of mass; a simulation mode that offers feedback on speed and maximum travel distance; a race mode that races the current design against other defined designs; a preview mode depicting the paper templates to be printed, and a print mode that provides feedback in the form of a printout. Figure 5.15 summarizes the feedback opportunities made available via these modes.

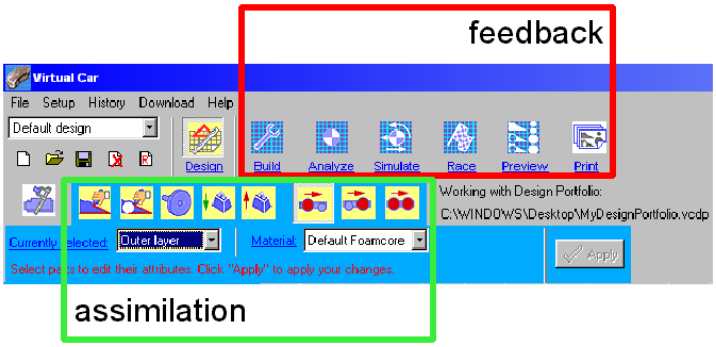


Figure 5.14. Interface elements involved with assimilation and feedback

5.3 Instrumentation

The goal of instrumentation was to record in a transparent manner a comprehensive history of user interaction such as parameter settings (i.e., assimilation) and performance tests (i.e., feedback requests). Codes were assigned to help track and record these activities.

Table 5-1. Design parameters and means of specification

Parameter	Value Type	Means of specification
Spring width	Numeric	Keystrokes
Spring length	Numeric	Keystrokes
Drive axle diameter	Numeric	Keystrokes
Other axle diameter	Numeric	Keystrokes
Drive wheel diameter	Numeric	Keystrokes
Other wheel diameter	Numeric	Keystrokes
Drive Axle Length	Numeric	Keystrokes
Other Axle Length	Numeric	Keystrokes
Spring-to-chamber-volume ratio	Numeric	Keystrokes
Friction drag force	Numeric	Keystrokes
Added weights	Numeric + Point	Keystrokes, mouse
Drive type (chamber location)	[Rear, Front, All]	Mouse click
Front wheel location	Point	Mouse click
Spring chamber style	[Circular, Octagonal]	Mouse click
Outer layer shape	Polygon	Mouse clicks
Inner layer shape	Polygon	Mouse clicks
Outer body layer material	Defined board material	List selection
Inner body layer material	Defined board material	List selection
Drive wheel material	Defined board material	List selection
Front wheel material	Defined board material	List selection
Drive axle material	Defined axle material	List selection
Front axle material	Defined axle material	List selection
Spring material	Defined spring material	List selection
Tread material	Defined tread material	List selection

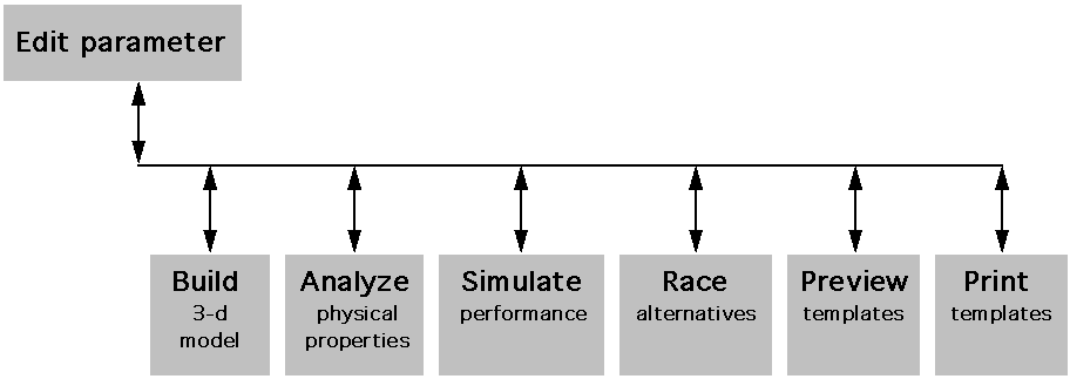


Figure 5.15. Feedback opportunities in Virtual Car

5.3.1 Codes for Design Parameters and Feedback Modes

Unique codes were assigned to each design parameter and feedback opportunity so that they may be recorded. These codes are shown in Tables 5-2 and 5-3.

Table 5-2. Codes assigned to design parameters

Code	Parameter	Abbreviation
1	Outer body layer material	OBMat
2	Inner body layer material	IBMat
3	Drive wheel material	DWMat
4	Front wheel material	FWMat
5	Drive axle material	DAMat
6	Front axle material	FAMat
7	Spring material	SpMat
8	Tread material	TrMat
9	Outer layer shape	OShape
10	Inner layer shape	IShape
11	Drive type	DType
13	Front wheel location	FWLoc
14	Spring width	SpWid
15	Spring length	SpLen
16	Drive axle diameter	DADia
17	Other axle diameter	OADia
18	Drive wheel diameter	DWDia
19	Other wheel diameter	OWDia
21	Drive Axle Length	DALen
20	Other Axle Length	OALen
22	Added weights	Weight
25	Spring to Case Area ratio	CaseRatio
28	Friction force	Friction
29	Case style	CStyle

Table 5-3. Codes assigned to assimilation events and feedback events

Code	Feedback level
0	Increment parameter value (type)
1	Exit parameter value (set)
2	Apply edited parameters
3	Build 3-D rendition
4	Check physical properties
5	Simulate performance
6	Race animation
7	Preview templates
8	Print templates
9	Built prototype (unimplemented)

5.3.2 The Design Portfolio File

As the designer uses the software, the codes corresponding to every parameter setting and feedback request that takes place are written to a file on the local hard drive, called the Design Portfolio file. The file also receives peripheral information such as the name of the car design, the date and time of each action, the parameter value that was set, and the last computed performance of the current design (travel time to finish line, maximum distance, and maximum speed). If the designer works with a single Design Portfolio file during the entire design process, a complete record of all user interaction with the software is recorded, even if it consists of multiple design sessions. Some example entries that might compose this type of file are shown in Table 5-4.

Table 5-4. Example excerpt of Design Portfolio file

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	11	FWD	-	-	-	1.07	15.90	18.80	0.47	MyDesign	11/27/01	12:01:24 PM	SetDrive	-	-	0	0	0	0	0	0
2	11	FWD	-	-	-	1.07	15.90	18.80	0.47	MyDesign	11/27/01	12:01:24 PM	DoDrive	-	-	0	0	0	0	0	0
1	11	AWD	-	-	-	1.07	15.90	18.80	0.47	MyDesign	11/27/01	12:01:27 PM	SetDrive	-	-	0	0	0	0	0	0
2	11	AWD	-	-	-	1.07	15.90	18.80	0.47	MyDesign	11/27/01	12:01:27 PM	DoDrive	-	-	0	0	0	0	0	0
1	11	RWD	-	-	-	1.07	15.90	18.80	0.47	MyDesign	11/27/01	12:01:31 PM	SetDrive	-	-	0	1	0	0	0	0
2	11	RWD	-	-	-	1.07	15.90	18.80	0.47	MyDesign	11/27/01	12:01:31 PM	DoDrive	-	-	0	1	0	0	0	0
0	14	1	-	-	-	1.07	15.90	18.80	0.47	MyDesign	11/27/01	12:01:51 PM	TypeSWid	-	-	-	-	-	-	-	-
0	14	1.	-	-	-	1.07	15.90	18.80	0.47	MyDesign	11/27/01	12:01:51 PM	TypeSWid	-	-	-	-	-	-	-	-
0	14	1.5	-	-	-	1.07	15.90	18.80	0.47	MyDesign	11/27/01	12:01:51 PM	TypeSWid	-	-	-	-	-	-	-	-
1	14	1.5	-	-	-	1.07	15.90	18.80	0.47	MyDesign	11/27/01	12:01:52 PM	ExitSWid	-	-	0	0	0	0	0	0
2	14	1.5	-	-	-	1.07	15.90	18.80	0.47	MyDesign	11/27/01	12:01:52 PM	DoSprWid	-	-	0	0	0	0	0	0
5	401	-	-	-	-	1.09	14.89	18.00	0.45	MyDesign	11/27/01	12:01:55 PM	VPDynamic	0	0	0	0	0	0	0	0
6	501	1	-	-	-	1.23	14.89	18.00	0.45	MyDesign	11/27/01	12:02:00 PM	VPRace	0	0	0	0	0	0	0	0

KEY:

Column	Contents
1	Assimilation/feedback identifier
2	Parameter code or feedback level code
3-6	Parameter values
7-10	Performance values
11	Car name
12-13	Date and time of event
14	Name of action
15-22	Advisory codes

Column 1 contains a code that identifies the action as either assimilation or feedback and also provides information about the level of involvement. Codes 0, 1, and 2

represent three components of an assimilation event: typing or clicking a parameter value, completing an edit of a parameter value, and formally applying the parameter value to the design. Code 0 indicates a single keystroke or other action that partially increments a parameter setting, such as when the user types one digit of a value into an edit field. Code 1 indicates that the user has completed entering a parameter value for a given parameter, either by tabbing into or clicking on a different edit field that represents a different parameter value, or by clicking on a button that leads to a feedback mode. Code 2 is generated whenever the user presses the Return key after editing a value or presses the "Apply" button, which is provided as a way of assuring the user that a value has been accepted. Codes 3, 4, 5, 6, 7, and 8 represent the six feedback modes that the Virtual Car software provides: Build, Analyze, Simulate, Race, Preview, and Print, respectively.

The code in Column 2 applies primarily to assimilation events, identifying the parameter that was edited according to the parameter codes shown in Table 5-2. If the event was a feedback request, this column contains an identifier for the feedback mode, ranging from 301 for the Build mode to 801 for the Print mode.

Columns 3 through 6 are reserved for the actual parameter value as represented in its edit field at the time of the event. Design parameters have a single value and require only one of these fields; the additional fields are reserved for future use.

Columns 7 through 10 report the last computed performance results at the time of the event. Column 7 reports the time in seconds to reach 15 feet; Column 8 reports top speed in feet per second; Column 9 reports powered distance in feet, and Column 10 reports the mass of the vehicle in pounds.

Columns 11 through 14 contain the current name of the car design, the date and time of the event, and an abbreviated name for the event to help in manually interpreting the file.

Columns 15 thru 22 are binary (0 or 1) codes reserved for advisories associated with the eight possible design error states that the Virtual Car software detects and reports to the user. Respectively, a value of 1 for Columns 15 through 22 indicates: car will break

traction (skid) on accelerating; car will not travel minimum distance of 15 feet; front axle is not connected to body; spring case is not connected to body; added weights are not connected to body; car body exceeds the maximum size for a printed template; spring cases of a four-wheel-drive car intersect each other; and portions of the body are floating freely from the main body.

5.4 Deploying the Instrumented *Virtual Car*

The instrumented version of Virtual Car was designed to be deployed transparently with minimal constraint on the design process of the subject. The primary aspect of data collection would involve the generation, maintenance, and collection of a Design Portfolio file unique to each subject. By asking each subject to maintain a single Design Portfolio file during his or her design process, data could be collected automatically and transparently as the subject used the software in the normal way.

Upon launching Virtual Car, the subject is prompted to either create a Design Portfolio (in the case of a new design), or to locate a preexisting Design Portfolio (in the case of continuing a design process). After the file has been created or located, activity is transparently recorded in the file until the user exits the software.

Once the subject has completed the design process, the Design Portfolio file must be collected. In order to most reliably identify the termination of the design process and match it to the true outcome of the process, the final car design should also be collected so that its performance characteristics may be compared to that reported in the Design Portfolio. Once the Design Portfolio files have been collected, analysis may proceed by extracting relevant information from the file and converting it to an A-F-DP timeline. For the purpose of the current study, specialized program routines were written to allow the loading of a Design Portfolio file and the generation of its A-F-DP timeline. This step in the analysis of the data is discussed further in Chapter 7.

6 Data Collection

The instrumented version of Virtual Car was deployed to collect data from volunteer subjects enrolled in two sections of Engineering 100 during Autumn Quarter 2001 and Winter Quarter 2002. This chapter describes the data collection procedures and the resultant data set.

6.1 Deployment of Instrument

Sections of Engineering 100 that teach the Virtual Car project typically hold the project during the second half of the quarter. At this time students have completed two other projects, a bridge design project and an engine dissection project [Kramlich and Fridley 1999, Safoutin et al. 2000]. The project begins with an introduction to the project and its design goals, physical principles governing the performance of vehicles, general strategies for achieving good performance, and a demonstration of the Virtual Car software. Several individual homework assignments are assigned, during what otherwise is primarily a group-based design effort.

During each of Autumn Quarter 2001 and Winter Quarter 2002, one section of Engineering 100 used the instrumented version of Virtual Car and acted as a source of design process data. Data collection took place by means of an individual homework assignment (Appendix A). This assignment asked each student to download the software and use it to develop a personal car design, followed by the answering of several essay questions about their experience in designing it. Each student was also asked to submit the Design Portfolio file and the file describing their final car design as evidence of their design effort. The files and essays were collected and graded by the instructor during the first two weeks of the project, and copies were retained for potential use in the study. At a point during the last week of each quarter, a consenting process was conducted. After the class was informed of the study, each student was given an opportunity to complete a consent form (Appendix B) by which they could confidentially grant or deny permission

for the data represented in their design files and essays to be incorporated into the study. The consent form was administered by a person unassociated with the study.

Following completion of the individual homework assignment, the project shifted into a group phase. Groups of four to five students collectively designed and built two to three physical prototypes over a three to four week period as is typical for the project. The work of each group was largely conducted during scheduled class hours, and consisted of a combination of work with the Virtual Car software, group discussion, and physical construction. Because the instrumented version of the software was in use, this activity tended to result in the creation of one or more Design Portfolio files residing on each workstation at the end of the project. These files would naturally contain a large number of entries, representing a mix of individual designing and interactive group designing. This group data was not directly included in the study, but was collected for possible use in developing and testing candidate measures and for ensuring that individual design files that were submitted as evidence of individual design activity were in fact unique from the group data collected on the workstation.

After each quarter was over, data was extracted from the individual data files and individual reflective papers that were collected. The instructor gathered the reflective essays and individual homework data files of consenting individuals. For possible reference purposes, the group data files of consenting groups were also collected from group workstations. The essay and individual data file of each consenting subject was then matched by name. For each pair of essay and data file corresponding to one individual, data was extracted into an independent data set that was labeled by a random code consisting of a number representing the individual and a letter representing the design team to which the individual belonged. This extraction process resulted in the names of individual students, group names, and all other direct or indirect identifiers being stripped and replaced with the random code.

The Fall and Winter quarter data collection episodes were conducted identically in all but one respect. In the Fall quarter, subjects were asked to create their personal design with the goal of achieving a "good balance" between speed and distance. In the Winter quarter, subjects were asked to design exclusively for maximum speed. Both groups

designed under the constraints that the car must not skid on acceleration and must travel at least 15 feet.

6.2 Data Collected

Two data sets were gathered, one from Autumn Quarter 2001 and another from Winter Quarter 2002. The raw data of each set consisted of three primary components as described below.

Individual Design Portfolio files

Each subject turned in the Design Portfolio file that was created during the process of designing the car for the homework assignment. These files contain the data representing the individual design process of each subject.

Individual Car Designs

Subjects also submitted a Virtual Car Design File (.vcar) describing their final design. These files were collected in order to confirm that the process described in the Design Portfolio matched the car that was actually designed.

Essay questions

Several essay questions were posed about the experience of designing the car. These questions were collected to provide information that might potentially be useful in interpreting the activity recorded in the design portfolio. The questions were assigned via an assignment web page linked from the course web page. The questions were the same in both quarters, with the exception of one question that pertained to the goal of the design process, which varied between quarters. The questions are outlined in Appendix A. This appendix also provides a facsimile of the assignment itself.

The raw data were then processed as described in the next section, forming two distinct data sets for analysis.

6.3 Data Preparation

Three primary issues were involved in preparing the raw data for analysis.

Consent

Of thirty-four students enrolled in Autumn Quarter 2001, two denied consent and five did not complete a consent form due to absence, leaving twenty-seven consenting subjects. Of thirty-six students enrolled in Winter Quarter 2002, two denied consent and seven did not complete a form, leaving twenty-seven consenting subjects.

Data Availability

Because the task was a graded homework assignment, students were naturally motivated to complete it for a grade but could not be required to do so. Accordingly some students who later provided consent had chosen not to complete the assignment, while others had failed to complete portions of it. In the Autumn quarter, five subjects did not submit a Design Portfolio file, leaving a total of twenty-two consenting subjects for which data was available. Furthermore, one of these subjects failed to provide answers to the essay questions, and another did not provide a self-rating of iterative character, leaving a total of twenty consenting subjects for which all data was available. In the Winter quarter, three did not turn in a Design Portfolio file, leaving a total of twenty-four consenting subjects with portfolio data. Additionally, two of these failed to report a self-rating of iterative character in their essay homework.

Data Validation

Some data from consenting subjects failed to be validated as representing the subject's actual effort. Several subject data sets from Autumn 2001 were set aside because the parameter values and performance estimates recorded in the Design Portfolio did not appear to match the car design that was actually turned in. This judgement was made in cases when either of the following was true: (a) terminal performance values recorded in the Design Portfolio did not match the performance of the submitted design, or (b) the

Design Portfolio was empty or did not record enough activity to have credibly developed the reported design.

Some data sets presented initial interpretive issues that were resolved by further analysis. All Design Portfolio files were inspected to determine whether the process was conducted in a single design session or over multiple sessions. Some files contained more than one session, typically including any of: (a) a short initial period in which the designer appeared to be exploring various software functions, without saving a design, (b) a later period during which the design was checked and printed but not edited, or only edited to explore some other options and then changed back again, (c) short sessions that took place during scheduled class time, included a variety of parameter edits but did not seem to result in progression of a car design, and did not lead to a saved design. Where necessary, the main design session was isolated from these undirected episodes by segmenting the Design Portfolio file into a main session file that was responsible for the submitted design and one or more others containing the contents of the undirected activity. The main session files were analyzed while the others were retained for context.

Several data sets were set aside because the submitted Design Portfolio was identical to the group portfolio that had accumulated on the group's assigned workstation, and not enough evidence could be found in either the submitted design or the portfolio file to determine which portion of the file represented the design process associated with the submitted design. These data sets were primarily in the Autumn 2001 data.

6.4 Resultant Data Sets

The final Autumn 2001 data set consists of sixteen matched sets of essays and design portfolios and four group data files. The Winter 2002 data set consists of twenty-four matched sets of essays and design portfolios. The two data sets are summarized in Tables 6-1 and 6-2.

These data sets form the basis of the data analysis that is conducted in Chapter 7.

Table 6-1. Summary of data from consenting subjects, Autumn 2001

	Code	Portfolio	Essay	Car file	Portfolio size (KB)	Number of feedback episodes
1	N/C	-	-	-	-	-
2	N/C	-	-	-	-	-
3	N/C	-	-	-	-	-
4	N/C	-	-	-	-	-
5	N/C	-	-	-	-	-
6	N/C	-	-	-	-	-
7	N/C	-	-	-	-	-
8	n/a	-	-	-	-	-
9	n/a	-	-	-	-	-
10	**A411**	*	*	*	**148**	**144**
11	**A433**	*	*	*	**56**	**27**
12	**B241**	*	*	*	**140**	**98**
13	**C695**	*	*	*	**28**	**15**
14	**X250**	*	*	*	**40**	**27**
15	**X397**	*	*	*	**52**	**45**
16	**Y318**	*	*	*	**60**	**58**
17	**Y389**	*	*	*	**44**	**35**
18	**Y402**	*	*	*	**140**	**61**
19	**Y418**	*	*	*	**64**	**32**
20	**K235**	*	*	*	**28**	**22**
21	**K238**	*	*	*	**68**	**68**
22	**K333**	*	*	*	**60**	**35**
23	**D477**	*	*	*	**40**	**26**
24	**W255**	*	*	*	**16**	**16**
25	**W455**	*	*	*	**24**	**14**
26	C629	x	*	*	36	-
27	D642	x	*	*	80,60	-
28	W462	x	*	*	280	-
29	W584	x	*	*	308	-
30	Y206	x	*	*	12	-
31	X703	x	n/a	*	152	-
32	X714	n/a	*	n/a	-	-
33	X815	n/a	*	n/a	-	-
34	D948	n/a	*	n/a	-	-
		16 validated portfolios	**24** essays	**22** cars		

Key:

N/C = no consent
n/a = not turned in
* = turned in and validated
x = turned in but not validated

Table 6-2. Summary of data from consenting subjects, Winter 2002

	Subject code	Portfolio	Essay	Car file	Portfolio size (KB)	Number of feedback episodes
1	N/C	-	-	-	-	-
2	N/C	-	-	-	-	-
3	N/C	-	-	-	-	-
4	N/C	-	-	-	-	-
5	N/C	-	-	-	-	-
6	N/C	-	-	-	-	-
7	N/C	-	-	-	-	-
8	N/C	-	-	-	-	-
9	N/C	-	-	-	-	-
10	**R246**	*	*	*	**69**	**55**
11	**R815**	*	*	*	**172**	**153**
12	**R819**	*	*	*	**47**	**13**
13	**S185**	*	*	*	**128**	**156**
14	**S368**	*	*	*	**56**	**86**
15	**S446**	*	*	*	**29**	**25**
16	**S527**	*	*	*	**223**	**228**
17	**S742**	*	*	*	**214**	**253**
18	**F234**	*	*	*	**13**	**7**
19	**F265**	*	*	*	**34**	**22**
20	**F336**	*	*	*	**62**	**73**
21	**L258**	*	*	*	**187**	**91**
22	**L564**	*	*	*	**71**	**81**
23	**U487**	*	*	*	**203**	**91**
24	**U750**	*	*	*	**168**	**140**
25	**G357**	*	*	*	**69**	**48**
26	**G365**	*	*	*	**68**	**52**
27	**G654**	*	*	*	**161**	**116**
28	**E150**	*	*	*	**101**	**117**
29	**E747**	*	*	*	**27**	**13**
30	**H245**	*	*	*	**169**	**152**
31	**H456**	*	*	*	**7**	**6**
32	**H837**	*	*	*	**38**	**13**
33	**U767**	*	*	*	**176**	**89**
34	F135	n/a	*	n/a	-	-
35	G773	n/a	*	n/a	-	-
36	F345	n/a	n/a	n/a	-	-
		24 validated portfolios	**26** essays	**23** cars		

Key:

N/C = no consent
n/a = not turned in
* = turned in and validated
x = turned in but not validated

7 Data Analysis

In this chapter, the feedback quality measure of iteration is applied to the data described in Chapter 6. The purpose of this application is to collect evidence regarding the validity and utility of the measure in a research application.

First, an approach is described for evaluating the validity of the measure by examining the results of its application. Next, general procedures for analyzing the data are described, followed by application of the measure to the data. The result of the application is then evaluated to gather evidence concerning the validity and utility of the measure.

7.1 Basis for Evaluating Validity

Evaluating the validity of a new measure can be a difficult task. Normally, one might seek to demonstrate convergent validity, by showing that the measure provides measurements that are consistent with those of known measures applied to the same data [Rosenthal and Rosnow 1984]. This approach is not applicable to the current study because proven measures of iterative character are not available for comparison. In this sort of situation, it is helpful to adopt the perspective that validity is not proven *per se* but is supported by a validity argument, which consists of supporting evidence and explanatory rationale that accumulates over time [Cronbach 1988]. The problem then becomes one of establishing a firm foundation for such an argument. A foundation may be begun by collecting evidence that the measure is applicable to real data, and that its measurements are consistent with expectations regarding the iterative character of processes to which it is applied.

In this study, the assigned task possesses several distinctive attributes that suggest several expectations regarding the iterative character of its solution process. First,

because the task is oriented toward a design goal that is not trivial to achieve, one would not expect subjects to conduct the design process at random but rather to direct it according to some intentional pattern that would be distinguishable from a random one. Second, because both groups were presented with a very similar design task, one would expect that the processes conducted by both groups would deviate from random in a similar way. Third, the assigned task might be expected to encourage a particularly iteration-rich solution process, because feedback about the performance of the evolving design is always available instantly at little cost. This contrasts it with other tasks that might impose a realistic cost on feedback and thereby discourage iteration.

If the data analysis shows that the measure has provided measurements consistent with these expectations, a foundation has been established for a validation argument upon which further studies may build. The next several sections investigate this question.

7.2 Analysis Procedures

To assist in processing of the data, data analysis functions were added to the research version of Virtual Car. Routines were implemented for loading a selected Design Portfolio file and displaying and printing its Assimilation-Feedback-Design Parameter (A-F-DP) timeline. The timeline is created by scanning the file for assimilation events (i.e., parameter edits) and feedback events (i.e., entry into a feedback mode), matching each assimilation event with its corresponding design parameter, and plotting both types of events in their relative temporal sequence. The routines were tested by applying them to selected Design Portfolio files from the Fall 2001 group and additional files that had been collected from group workstations. The data sets from Fall 2001 and Winter 2002 were then processed by use of these routines, resulting in two printed sets of timelines.

For each subject, the discrete feedback quality measure was then applied by manual examination of parameter quantity and parameter identity patterns according to the procedure described in Chapter 4. Issues relating to correct and consistent interpretation of the timeline data were addressed prior to analysis and are summarized in Appendix C.

Sample timelines for two subjects are shown in Figure 7.1. Each timeline is continued on a second line. The upper timeline is that of Subject A368 (Fall quarter), and consists of a total of 85 feedback episodes. The lower timeline is that of Subject Y318 (Winter quarter), and consists of 57 feedback episodes. On the left edge of each timeline, the abbreviated name of each edited design parameter is shown in order of visitation (abbreviations are as shown in Table 5-2). Each feedback event is indicated by a short vertical line, while each assimilation event is indicated by a series of rectangles and/or circles. Narrow rectangles indicate individual keystrokes as a number is typed, or individual clicks as a shape is drawn. A wider rectangle or a circle indicates that the edit has been completed. The circle is generated if the user pressed the "Apply" button or hit Return to signal completion of an edit operation prior to entering a feedback mode. This action has no design function but is the typical way by which a designer ends an edit operation.

Each feedback episode in the figure has been annotated with a letter (A, B, C, or D) to indicate its feedback quality classification. The process of classifying each feedback event in this manner is the primary manual act in coding a timeline. The letters may then be entered into a spreadsheet in order to calculate percentages A, B, C, and D that form the basis of the feedback quality profile of the process.

After the timeline analysis was completed, a small sample of timelines were re-analyzed to check for mistakes. It is important to remember that application of the feedback quality measure to a timeline representation does not require subjective judgement on the part of the analyst; human error was the primary concern here. The analysis of this sample did not reveal any significant mistakes.

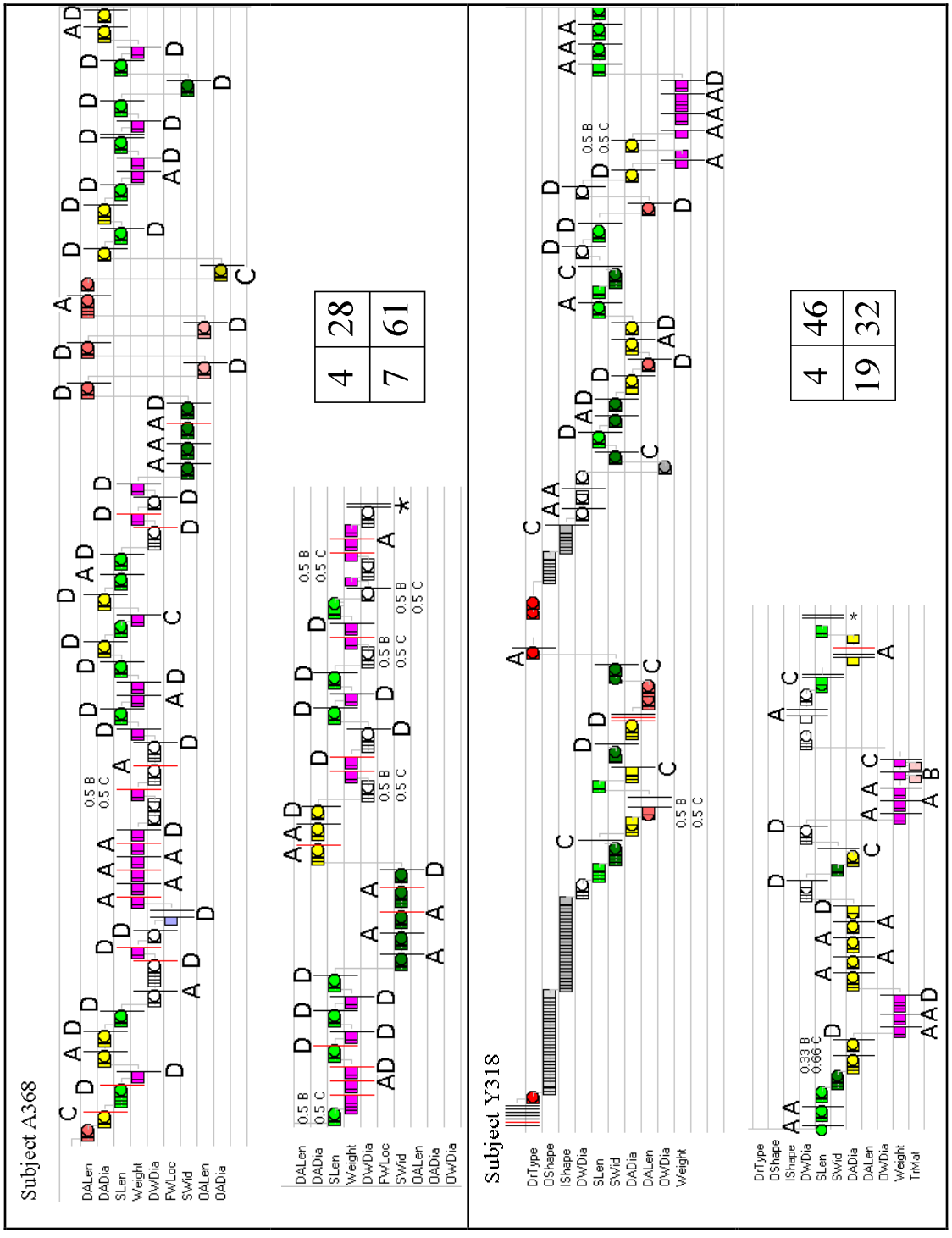


Figure 7.1. Sample A-F-DP timelines for subjects A368 and Y318

7.3 Description of Results

7.3.1 Discrete Feedback Quality

The result of the feedback quality analysis is summarized in Table 7-1. Four percentages for the feedback patterns A, B, C, and D are shown for each process, along with their means, standard deviations, and range.

Table 7-1. Feedback quality measurements, Fall 2001 and Winter 2002

Fall 2001					Winter 2002				
Subject	% A	% B	% C	% D	Subject	% A	% B	% C	% D
K235	14	0	19	67	A185	45	7	15	31
K238	63	3	7	27	A368	28	4	7	61
K333	14	7	38	40	A446	42	6	10	42
A411	56	4	11	30	A527	50	2	11	37
A433	50	1	22	27	A742	25	9	30	36
B241	45	7	20	28	F234	33	8	42	17
C695	29	24	40	7	F265	48	6	13	33
D477	40	9	19	32	F336	63	1	9	28
W255	60	0	7	33	R246	44	5	21	30
X250	23	7	28	42	R815	46	5	13	36
X397	25	7	23	45	R819	0	45	39	17
W455	8	8	54	31	L258	46	9	17	29
Y318	46	4	19	32	L564	60	8	7	25
Y389	56	11	19	15	U487	40	7	17	36
Y402	35	8	14	43	U750	47	5	14	34
Y418	32	9	17	42	U767	49	4	6	41
					G357	21	9	27	43
					G365	45	3	13	39
					G654	45	16	24	15
					E150	55	5	8	32
					E747	25	0	25	50
					H245	33	5	20	42
					H456	40	7	13	40
					H837	8	25	41	25
Mean	**37.2**	**6.8**	**22.3**	**33.8**	**Mean**	**39.1**	**8.4**	**18.4**	**34.1**
Std dev	17.4	5.7	12.5	13.5	**Std dev**	15.1	9.3	10.8	10.5
Min/Max	8/63	0/24	7/54	7/67	**Min/Max**	0/63	0/45	6/42	15/61

Table 7-2 expresses the data of Table 7-1 in the form of feedback quality profiles. Percentages for A, B, C, and D are printed in the upper-right, upper-left, lower-left, and lower-right quadrants respectively.

Table 7-2. Feedback quality profiles, Fall 2001 and Winter 2002

%B	%A
%C	%D

Key

Fall 2001				Winter 2002					
K235		W255		A185		R246		G357	
0	14	0	60	7	45	5	44	9	21
19	67	7	33	15	31	21	30	27	43
K238		X250		A368		R815		G365	
3	63	7	23	4	28	5	46	3	45
7	27	28	42	7	61	13	36	13	39
K333		Y418		A446		R819		G654	
7	14	9	32	6	42	45	0	16	45
38	40	17	42	10	42	39	17	24	15
A411		X397		A527		L258		E150	
4	56	7	25	2	50	9	46	5	55
11	30	23	45	11	37	17	29	8	32
A433		W455		A742		L564		E747	
1	50	8	8	9	25	8	60	0	25
22	27	54	31	30	36	7	25	25	50
B241		Y318		F234		U487		H245	
7	45	4	46	8	33	7	40	5	33
20	28	19	32	42	17	17	36	20	42
C695		Y389		F265		U750		H456	
24	29	11	56	6	48	5	47	7	40
40	7	19	15	13	33	14	34	13	40
D477		Y402		F336		U767		H837	
9	40	8	35	1	63	4	49	25	8
19	32	14	43	9	28	6	41	41	25

7.3.2 Categorization Based on Discrete Feedback Quality

Tables 7-3 and 7-4 assign each process to its nearest canonical category by manual application of the *X+y* naming convention as described in Chapter 4. Also as described in that chapter, the categories are ranked in order of their relative feedback quality rank. The rank 1 represents the best quality, corresponding to a Type A-dominant process.

Table 7-3. Feedback quality classification, Fall 2001

Rank	1	2		3	4		5				6		7	8		9
Category	A	A+d	A+b	A+c	D+a	B+a	D	B	D+b	B+d	D+c	B+c	C+a	C+d	C+b	C
Members		K238		Y389	X397						K235		C695	W455		
		A411			Y402						K333					
		A433			Y418						X250					
		B241														
		D477														
		W255														
		Y318														
Count	0	7	0	1	3	0	0	0	0	0	3	0	1	1	0	0
Percentage	50%				37.5%								12.5%			

Table 7-4. Feedback quality classification, Winter 2002

Rank	1	2		3	4		5				6		7	8		9
Category	A	A+d	A+b	A+c	D+a	B+a	D	B	D+b	B+d	D+c	B+c	C+a	C+d	C+b	C
Members		A185		G654	A368						A742	R819	F234	H837		
		A527			A446						G357					
		F265			H456											
		F336			E747											
		R246														
		R815														
		L258														
		L564														
		U487														
		U750														
		U767														
		G365														
		E150														
		H245														
Count	0	14	0	1	0	0	4	0	0	0	2	1	1	1	0	0
Percentage	62.5%				29.2%								8.3%			

Referring to Table 7-3, it may be seen that the Fall 2001 data was predominantly classified A-dominant, with a significant showing of D-dominance. Only two processes

deviated from this trend by exhibiting a C+a or C+d pattern. In all, only six of the sixteen canonical categories were occupied, with the greatest absences in B-dominant and C-dominant categories. In the Winter 2002 data depicted in Table 7-4, the picture is much the same. The great majority of processes were classified as A+d, with other smaller showings in primarily D- and C-dominant categories.

Tables 7-5 and 7-6 aggregate the canonical categories into several larger categories representing the most dominant quadrant. The D- and B-dominant categories are merged into a single category to reflect their similar feedback quality ranks. As shown in both tables, processes in both groups were primarily A-dominant, ranging from 50% in Fall 2001 to 62.5% in Winter 2002. The B and D population amounted to 37.5% in the Fall group and 29.2% in Winter. C-dominant processes were the smallest category in both groups, comprising 12.5% in Fall and 8.3% in Winter.

Table 7-5. Comparison of percentages in major categories, Fall and Winter

Group	A-dominant	D- or B-dominant	C-dominant	*total*
Fall 2001	50.0 %	37.5 %	12.5 %	100%
Winter 2002	62.5 %	29.2 %	8.3 %	100%

Table 7-6. Comparison of frequencies in major categories, Fall and Winter

Group	A-dominant	D- or B-dominant	C-dominant	*n*
Fall 2001	8	6	2	16
Winter 2002	15	9	2	24

7.4 Comparison of Results to Expectations

These feedback quality measurements are now related to expectations regarding the iterative character of the observed processes.

7.4.1 Expectation 1: Both Groups Design Nonrandomly

If the design task does not favor a random solution process, the measurements would be expected to indicate that both groups designed nonrandomly. Demonstrating this result would consist of showing that measurements of both groups are statistically distinct from the measurements expected if the processes were random.

The issue of defining a "random" design process is a fairly complex one. A precise definition requires that several judgements be made regarding the number of design parameters that the designer considers and how the process is conducted. This issue is addressed in Appendix F. In this analysis, we will make the assumption that a random process results in an equal likelihood that a feedback episode will be classified as Type A, B, C, or D. The discussion in Appendix F suggests that this assumption is reasonable for a problem having a small number of design parameters, and for the evaluative purpose at hand it represents a more conservative assumption than one that attempts to account for the number of parameters that were available to the subjects in this study.

In the Fall data set (n=16), eight processes were categorized as Type A, six as either Type B or D, and two as Type C. If each type were equally likely, the expected distribution would have been four A, eight B/D, and four C. These cases are compared in Table 7-7.

Table 7-7. Observed distribution vs. random expectation, Fall 2001

Fall Qtr (n=16)	A	B,D	C	
Expected (random)	4	8	4	16
Observed	8	6	2	16
	12	14	6	

$\chi^2 = 5.5$, $p = 0.064$

Fisher exact test: $p = 0.38$

By inspection, the distribution of Fall processes appears to differ from random. A significance test based on the chi-squared statistic indicates a 6.4% chance that this variation could result from random chance.

Because two of the expected values are smaller than 5, the suitability of the chi-squared statistic as a measure of significance is called into question. The Fisher exact test, which is considered preferable to the chi-squared test [Zar 1996] and is valid for small sample sizes, was employed as an alternative. This test indicates a 38% probability that the observed effect could result from random variation. The small size of the Fall sample (n = 16) makes it difficult to judge the significance of this variation by either test.

The Winter data set had a larger sample size (n = 24). Fifteen processes were categorized as Type A, seven as either Type B or D, and two as Type C. The expected distribution for a random process would have been six A, twelve B/D, and six C. These cases are compared in Table 7-8.

Table 7-8. Observed distribution vs. random expectation, Winter 2002

Winter Qtr (n=24)	A	B,D	C	
Expected (random)	6	12	6	24
Observed	15	7	2	24
	21	19	8	

$\chi^2 = 18.2$, p = 0

Fisher exact test: p = 0.03

By inspection, it is apparent that the distribution of Winter processes differs from random. Here, the chi-squared statistic indicates that the variation from random is significant (p = 0). The Fisher exact test provides a similar conclusion (p = 0.03).

The problem presented by the small size of the Fall sample might be alleviated by merging the Fall and Winter groups together. This is statistically appropriate only if there is reason to believe that the two groups are similar. This issue will be revisited in a later section after the similarity of the two groups has been examined.

7.4.2 Expectation 2: Both Groups Design Similarly

Because the task presented to both groups was similar, one would expect that both groups would differ from random in a similar way. The measure would thus gain additional support if its measurements indicate that the two groups are similarly distributed among Types A, B, C, and D to a degree that is statistically significant. this can be evaluated by comparing the distribution of Fall data to that of Winter.

By inspection, it is apparent that both groups are dominant in Type A and favor Types B and D secondarily. Is this similarity significant? If so, one would expect that a significance test would return a relatively large *p* value, indicating a lack of significant difference between the two groups.

In comparing Winter to Fall, the Fall data set becomes the expected case and is converted to a set of expected values based on a sample size n=24 as shown in Table 7-9.

Table 7-9. Comparison of Winter to Fall (expected values)

	A	B,D	C	
Expected (Fall)	12	9	3	24
Winter	15	7	2	24
	27	16	5	

$\chi^2 = 1.53$, p = 0.47

Fisher exact test: p = 0.70

The chi-squared test indicates that the two groups are not significantly different (p = 0.47). The Fisher exact test delivers a similar conclusion (p = 0.70). These results suggest that the measurements are consistent with our expectation that Winter and Fall should be quite similar in their distribution among each category.

7.4.3 Expectation 3: Both Groups Favor Type A Iteration

We now examine the nature of the similarity between the two groups. It may be argued that the Virtual Car-based design task encourages a particularly Type A and Type D process. Because feedback is available at any time with little penalty in terms of cost or time, single variables may be submitted for feedback as easily as multiple variables. Given the higher quality of feedback in the former case, Type A and D iteration should be favored if the process is nonrandom. Furthermore, designing for speed largely involves repeated optimization of several individual parameters that control a tradeoff between propulsive force and traction. This iterative optimization process represents Type A iteration. One would therefore expect the measurements to show that both groups were similar in being distributed specifically toward Type A iteration.

Table 7-10 compares the A-dominance of the Fall group (A = 8, other = 8) to the expected values for a random process (A = 4, other = 12). The chi-squared test indicates that the Fall group differs significantly from the random case (p = 0.02). However, the Fisher exact test yields a less optimistic outcome (p = 0.27).

Table 7-10. Comparison of Fall A-dominance to random expectation

Fall 2001	A-dominant	other	
Expected (random)	4	12	16
Observed	8	8	16
	12	20	

$\chi^2 = 5.33$, p = 0.02

Fisher exact test: p = 0.27

Table 7-11 depicts the analysis for the Winter group. Both the chi-squared and Fisher exact tests indicate that the Winter group is significantly different from random with regard to its Type-A dominance.

Table 7-11. Comparison of Winter A-dominance to random expectation

Winter 2002	A-dominant	other	
Expected (random)	6	18	24
Observed	15	9	24
	21	27	

$\chi^2 = 18.0$, p = 0

Fisher exact test: p = 0.02

These results suggest that the measurements are generally consistent with our expectations that both groups should favor Type A iteration. The uncertainty of this conclusion with respect to the Fall group results in part from the small size of the Fall sample, and might be addressed by combining the groups. This analysis is performed in the next section.

7.4.4 Effect of Combining Groups

With regard to Expectations 1 and 3, it was difficult to demonstrate the significance of the effect in the Fall group because its small sample size does not provide sufficient statistical power. However, the analysis of Expectation 2 demonstrated that the two groups are not significantly different. This allows us to re-examine Expectations 1 and 3 by combining the Fall and Winter data. When combined, the two groups create a sample size of 40 processes that should be sufficient to determine significance.

Table 7-12 shows the result of significance tests on the combined data with respect to Expectation 1. According to both tests, the combined group varies significantly from the expected random case.

Table 7-12. Observed distribution vs. random expectation, combined

Fall+Winter (n=40)	A	B,D	C	
Expected (random)	10	20	10	40
Observed	23	13	4	40
	33	33	14	

$\chi^2 = 23.0$, p = 0

Fisher exact test: p = 0.01

Table 7-13 shows the result of significance tests on the combined data with respect to Expectation 3. Again, both the chi-squared and the Fisher exact test indicate that the combined group is significantly A-dominant.

Table 7-13. Comparison of combined Winter/Fall A-dominance to random expectation

Fall+Winter	A-dominant	other	
Expected	10	30	40
Observed	23	17	40
	33	47	

$\chi^2 = 22.5$, $p = 0.00$

Fisher exact test: $p = 0.01$

These observations serve to strengthen the case that the measurements are consistent with Expectation 1 and Expectation 3.

7.5 Summary

This analysis has provided evidence that the measure has yielded measurements that agree well with expectations dictated by the nature of the design task.

Expectation 1 suggested that the measure should indicate that designers did not design randomly. Under a particularly conservative definition of a random process (as compared to alternative definitions outlined in Appendix F), the measure does indicate that the Winter group as well as the combined Fall and Winter group vary significantly from random. The statistical power provided by the Fall data alone is too limited to draw this conclusion with confidence, although the observed effect does point in the right direction. In all cases, the conclusion would be strengthened if any of the alternative definitions of a random process were to be employed instead.

Expectation 2 suggested that the measure should indicate that both the Fall and Winter groups designed in a similar way. The measurements indicate that the Fall and Winter groups are not significantly different, which is consistent with this expectation.

Because the satisfaction of Expectation 2 supports the similarity of the two groups, it becomes possible to group the Fall and Winter data together to improve the statistical power of the sample. The combined data provides additional support for the conclusion that the measurements were consistent with Expectation 1.

Expectation 3 suggested that the measure should indicate that both groups similarly favored Type A iteration. Measurement of the Winter group is strongly consistent with this expectation. Again, the Fall group alone lacks sufficient statistical power to draw a this conclusion with confidence, but when combined with the Winter group, the conclusion is strengthened.

These results suggest that the feedback quality measure of iteration has been successful in this application, and demonstrates that it is sufficiently valid to express similarity and difference among individual design processes in this example application. The result of this application appears sufficient to encourage further trials of the measure, through which additional evidence for validity may be accumulated.

8 Conclusions

This study was concerned with the question,

> *How may one objectively draw distinctions among observed design processes in terms of relevant aspects of their iterative character?*

It has sought to develop, apply, and validate a measure of iteration that would facilitate empirical research into the role of iteration in design and its influence on design outcome.

A conceptual framework of design iteration led to the recognition of a particularly interesting form of iteration, known as feedback iteration. A model of design was advanced to suggest several timeline formats for representation of an observed design process in terms of feedback activity. One timeline format was found to express two types of feedback patterns suggestive of an influence on outcome. These patterns formed the basis of a measure known as the discrete feedback quality measure of iterative character. An instrument was then developed for collecting data sufficient to apply the measure. Two sets of data were collected representing design processes of novice designers. Individual processes were then differentiated in terms of their iterative character by applying the feedback quality measure.

This chapter reviews the findings of this study with regard to the validity of the measure, its utility in design research, its application to other design tasks, and future work that might employ or improve the measure.

8.1 Validity of Measure

The study has demonstrated that the measure may be successfully applied to empirical data, and that the measurements it provides in this example application are consistent with what would be expected given the nature of the design task and the

subjects that performed the processes. This leads to the conclusion that the measure was effective at measuring real differences in the iterative character of the processes to which it was applied. Specifically:

(1) The measure indicates that both groups designed in a nonrandom manner as expected.

(2) The measure indicates that both groups designed similarly, consistent with the expectation resulting from the similarity of their design task.

(3) The measure indicates that both groups similarly favored Type A iteration, as expected due to the nature of the design task.

These results provide initial support for the validity of the feedback quality measure of iteration, and encourage further trials of the measure.

8.2 Utility of Measure

The study has shown that the measure makes it possible to draw distinctions among observed design processes in terms of concrete metrics relating to their iterative character. Previous approaches to measurement of iteration have relied on labor-intensive methods that require human judgement, such as verbal protocol methods. This measure, when applied to a parametric design task that has been appropriately instrumented, reduces the need for human judgement. Empirical measurement of iterative character may therefore be achieved at a much lower cost than that of previous methods.

Because the measure can be implemented in an automatic way, the measure could potentially be applied in real-time, as a design process takes place. For example, the measure may be integrated into a professional virtual prototyping system to provide feedback to the designer regarding the iterative character of his or her process as it unfolds. Alternatively, the measure could be made available to a student designer who is

carrying out a design process in an educational setting. The availability of such real-time feedback could potentially serve as a means to stimulate introspection about the design process for purposes relating either to performance or education.

8.3 Application to Other Design Tasks

Other Parametric Tasks

The data in this study was collected by applying an instrumentation procedure to a parametrically-structured, interactively-performed design task. This type of design task was selected in part due to practical considerations, such as the preference for a task that is rich in feedback iteration, the potential to instrument it for automatic data collection, and its accessibility for instrumentation. These considerations are potentially relevant to anyone who seeks a design task that may be studied with respect to its iterative character. It follows that others might seek to study other design tasks of this type. If the candidate task has an explicitly parametric structure, and assimilation and feedback activity is interactive and can be instrumented, it should be possible to base a data collection instrument upon the task by applying an instrumentation approach similar to that described in Chapter 5.

Tasks that are structured parametrically but not interactively performed might potentially be employed as a source of data as well. An example of such a task would be any conventional design problem that has been framed in terms of design parameters and functional requirements, such as those described by Suh [1990]. For these tasks, design parameters are well defined and should be easy to identify. However, the lack of an interactive implementation means that assimilation and feedback activity cannot be captured automatically through direct instrumentation of parameter edits and feedback requests. This activity must be manually identified through more traditional methods such as verbal protocols. This would require the development and application of a coding scheme to identify this activity, and verification of the result by checking the interrater reliability of multiple coders. Once assimilation and feedback have been isolated in this

manner, it should be straightforward to plot these events on a timeline and create feedback quality profiles.

Non-parametric Tasks

The ability to measure iterative character in a parametrically structured design task is a significant development in its own regard. However, the applicability of the measure to non-parametric tasks is an important issue as well. Many studies have presented subjects with unstructured tasks that could be characterized as non-parametric tasks; for example, the conceptual design of a playground [Atman et al. 1996], [Atman and Bursic 1998] or the design of a remote controlled robot that transports items between two points [Valkenburg and Dorst 1998]. The feedback quality measure of iteration would have greater utility if it could be applied to relatively unstructured design tasks such as these, as a supplement to other process measures that can be applied to these tasks.

Applying the feedback quality measure involves the measurement of two key patterns: the number of distinct modifications that have taken place each time the design is evaluated (parameter quantity), and the repetition of modifications to specific aspects of the design (parameter identity). A parametric structure provides a framework that facilitates the identification of specific aspects of the design and their modifications. Absent this structure, these features may be more difficult to recognize.

As suggested by Suh [1990], most if not all design problems may potentially be understood in terms of reasonably distinct design parameters and functional requirements. One approach to applying the measure to non-parametric design activity might begin by analyzing protocol data for evidence of design parameters and functional requirements that are implicitly being considered by the designer. Virtually any design task is likely to elicit statements that concern these fundamental elements of the problem, whether or not they have been formalized in the problem structure. Once recognized, they may then act as pointers to assimilation and feedback activity, just as in a task that is parametrically structured.

Verbal protocols are a particularly rich data source from which many specific types of information may be extracted. The information necessary to detect design parameters, functional requirements, and assimilation or feedback activity is not fundamentally different from that which is typically collected in verbal protocol studies. Several verbal protocol studies have already demonstrated the use of coding categories that isolate information of this sort.

As one example, consider the playground design task of Atman and Bursic [1998]. Verbal content was characterized according to four descriptive categories: "design step", "information processed", "activity", and "object". In particular, "information processed" represented the general topic that the subject was addressing (if any), such as budget, material costs, or safety. "Object" represented the physical component of the artifact that the information referred to (if any), such as a swing set, a slide, or landscaping. If the subject were to make the statement "we could build a slide out of two or three sheets of plywood for about twenty dollars", the information processed would be coded as *material cost*, and the object coded as *slide*.

This statement could additionally be coded with respect to whether it represents *assimilation* or *feedback* (if either), and which *design parameter* it concerns (if any). The various physical components of the emerging playground, such as its play equipment and the site, may be considered design parameters because they are what the designer modifies in order to fulfill the functional requirements of the playground. Thus the "object" category has the effect of identifying design parameters. The "information processed" could be inspected further to identify specific content as evidence of assimilation or feedback. A statement could be coded as assimilation if it contains information that advances the design state, or as feedback if it evaluates the current design state. The example statement above would then be coded as *assimilation*, with respect to the design parameter *slide*. That is, it advances the state of the design by proposing an inexpensive plywood slide. Any subsequent statements that act to evaluate this proposal would be coded as feedback. For example, the statement "a plywood slide might give people splinters -- better not do that" would be coded as feedback, because it represents an evaluation of the current design state.

Several other studies have successfully applied similar coding categories to a variety of design tasks. A verbal protocol study by Gero and McNeill [1998] observed designers conducting the design of a bicycle luggage rack. Their set of coding categories included "proposing a solution" (e.g., "the way to solve that is...") and "evaluating a proposed solution" (e.g., "this solution is faster and cheaper than..."). Statements assigned to these categories could alternatively be coded as assimilation and feedback, respectively. Other studies of electrical circuit design and mechanism design have employed categories such as "assimilate", "specify", and "verify" [Ullman et al. 1988], [Stauffer and Ullman 1991], which similarly relate to assimilation and feedback. The fact that these coding categories were successfully applied to a diverse group of design tasks suggests that it is feasible to develop a coding scheme that allows the feedback quality measure to be applied to non-parametric design tasks.

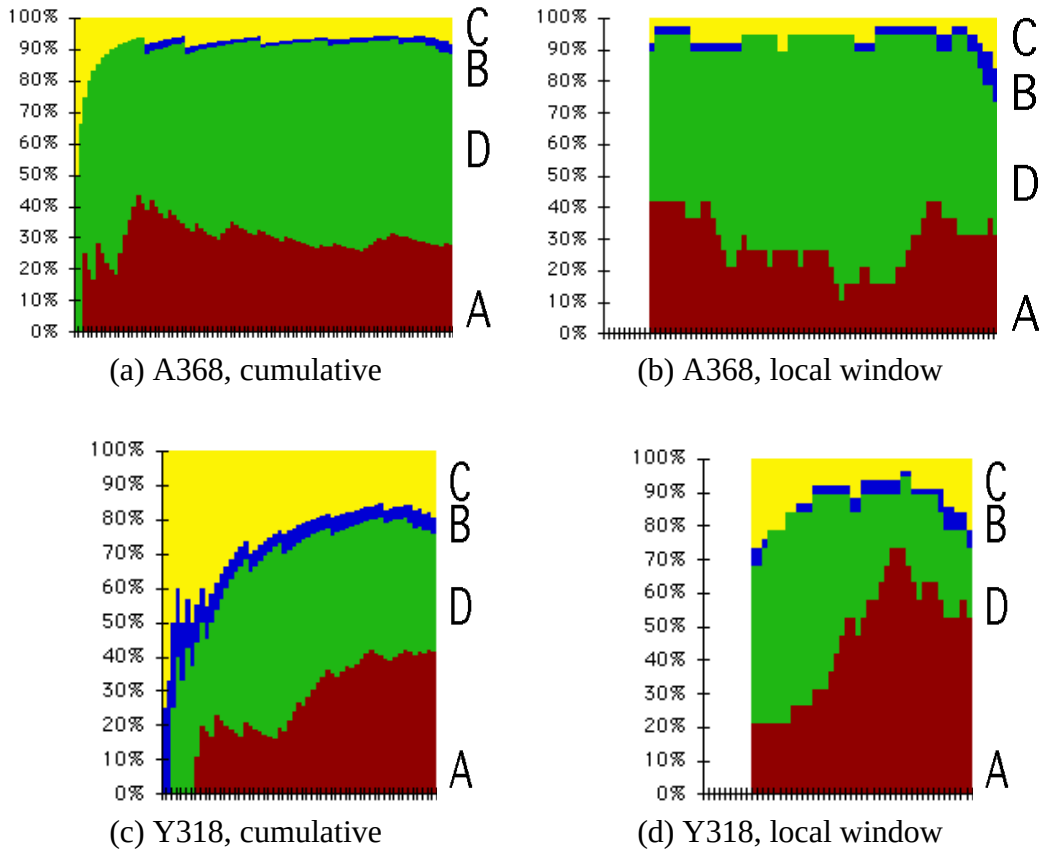
8.4 Future Work

In light of these results, several opportunities for additional research are suggested.

Additional Measures

One area concerns the further investigation of other measures of iteration that are related to the feedback quality measure.

As implemented in this study, the feedback quality measure represents an average pattern distribution for an entire process. It does not express local variations in pattern distribution during the process. As shown in Figure 8.1, the pattern distribution of a design process can vary substantially as the process unfolds. The variation may be depicted cumulatively as in Figure 8.1(a) and 8.1(c), or locally by tracking a moving window of a fixed number of events, as shown in Figure 8.1(b) and 8.1(d). Such variations might have utility in relating a process to other processes, or to design outcome. Future work will investigate this issue.



(a) A368, cumulative (b) A368, local window

(c) Y318, cumulative (d) Y318, local window

Figure 8.1. Examples of within-process variation in feedback quality

The *iterativity ratio*, discussed in Appendix D, is a simple measure of relative reliance on assimilation and feedback. It carries interesting implications that suggest utility of the measure in design research. In a design situation in which the cost of obtaining feedback is relatively high, one would expect a relatively low iterativity value because a natural reaction would be to submit more parameter changes per feedback request. On the other hand, in a design environment in which synthesis of design decisions is difficult, costly, or unreliable, one might expect the designer to rely more heavily on analysis through feedback, causing the process to have a relatively high iterativity value. In an experimental setting, one might explore this relationship by manipulating these variables, observing their effect on iterativity, and relating iterativity to design outcome.

The *computed* version of the feedback quality measure, discussed in Appendix E, is an alternative to the discrete feedback quality measure that was applied in the current

study. The computed measure accounts for parameter quantity more accurately by taking into account the actual number of parameters that act to confound a feedback event. The computed measure also allows depiction of feedback quality as a point on a plane rather than as a quad of four percentages. This opens the possibility of depicting many processes on a scatter plot and clustering the points visually or analytically in order to achieve similar groupings. The utility of this measure relative to the discrete measure remains to be investigated.

Applications

Useful applications of the feedback quality measure are another potential subject for further work. As foreshadowed previously, real-time implementation of the measure during the execution of a design process suggests that a program of research might be conducted to investigate the educational or professional utility of the measure. The measure might also be applied retrospectively to previously gathered verbal protocol data so that the results may be compared to other measures originally derived from the protocol analysis. This would call for the further development of reliable coding procedures for applying the measure to non-parametric design problems.

Finally, application of the measures to parametric design problems other than the one employed in this study remains to be demonstrated. This study has demonstrated the application of the measure to a parametrically structured, interactively implemented design problem. While the conditions for application of the measure to this type of problem are well established, the measure has yet to be applied outside of an interactive parametric environment.

Design researchers are continually seeking authentic design problems that may be studied effectively in an experimental setting. By itself, the capability demonstrated here for the measurement of iterative character in parametrically-structured, interactive design tasks amounts to a significant development for design research. It provides a systematic method to gather data from a rapidly growing variety of design environments, many of which are accessible to novice subjects. Parameterizing subject design problems and

moving them to instrumented interactive computer based environments for study purposes could represent a new approach to the empirical study of design processes, providing an alternative to verbal protocol analysis at a dramatically lower cost.

8.5 Conclusion

In summary, this study has provided a new and unique measure of design iteration that may be employed in empirical research regarding influences of iterative character on design outcome. The measure has been shown to be applicable to empirical data, has provided results that were effective and replicable in an example application, and can be implemented at a relatively low cost. These results suggest a strong potential for similar success in a variety of similar research applications.

References

Adam, J. A., "Virtual Reality is for Real", *IEEE Spectrum*, v30 n10, pp. 22-29 (1993).

Adams, R., "Cognitive Processes in Iterative Design Behavior", Doctoral Dissertation, College of Education, University of Washington (2001).

Adams, R., J. Turns and C. J. Atman, "Educating Effective Engineering Designers: The Role of Reflective Practice", *Designing in Context - Design Thinking Research Symposium 5*, December 18-20, Delft, The Netherlands (2001).

Afifi, A. A. and V. Clark, Computer-Aided Multivariate Analysis. Wadsworth Inc., Belmont, California (1984).

Ahmadi, R. and H. Wang, "Rationalizing Product Design Development Processes", Working Paper, Anderson Graduate School of Management, UCLA (1994).

Atman, C. J. and K. M. Bursic, "Teaching Engineering Design: Can Reading a Textbook Make a Difference?", *Research in Engineering Design*, v8 pp. 240-250 (1996).

Atman, C. J., K. M. Bursic, and S. L. Lazito, "An Application of Protocol Analysis to the Engineering Design Process", *Proceedings of the 1996 ASEE Annual Conference*, Session 2530 (1996).

Atman, C. J. and K. M. Bursic, "Verbal Protocol Analysis as a Method to Document Engineering Student Design Processes", *Journal of Engineering Education*, pp. 121-132 (1998).

Atman, C. J., J. R. Chimka, K. M. Bursic, and H. L. Nachtmann, "A Comparison of Freshman and Senior Engineering Design Processes", *Design Studies*, v20 n2, pp. 131-152 (1999).

Austin, S., J. Steele, S. Macmillan, P. Kirby and R. Spence, "Mapping the Conceptual Design Activity of Interdisciplinary Teams", *Design Studies*, v22 pp. 211-232 (2001).

Ball, L. J., J. St. B. T. Evans and I. Dennis, "Cognitive Processes in Engineering Design: A Longitudinal Study", *Ergonomics*, v37 n11, pp. 1753-1786 (1994).

Ball, L. J., J. St. B. T. Evans, I. Dennis and T. C. Ormerod, "Problem-solving Strategies and Expertise in Engineering Design", *Thinking and Reasoning*, v3 n4, pp. 247-270 (1997).

Banares-Alcantara, R., "Design Support Systems for Process Engineering - I. Requirements and Proposed Solutions for a Design Process Representation", *Computers in Chemical Engineering*, v19 n3, pp. 267-277 (1995).

Beakley, G. C, D. L. Evans, and J. B. Keats, Engineering: An Introduction to a Creative Profession, 5th ed. New York: MacMillan Publishing Company (1986).

Beckert, B. A., "Venturing Into Virtual Product Development," *Computer-Aided Engineering*, v15 n5, pp. 45-50 (1996).

Blanchard, D., "Ford Turns to Virtual Prototyping for Concurrent Engineering", *Intelligent Manufacturing*, v2 n10, p. 23 (1996).

Blandford, S. and R. P. Hope, "Systematic Methods for the Problem Solving Process With Particular Reference to Design", *Proceedings of the IEEE*, Part A, v132, pp. 199-212 (1985).

Bodker, S., "Understanding Representation in Design", *Human Computer Interaction*, v13 pp. 107-125 (1998).

Braha, D. and O. Maimon, "The Design Process: Properties, Paradigms, and Structure", *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, v27 n2, pp. 146-166 (1997).

Braha, D. and O. Maimon, "The Measurement of a Design Structural and Functional Complexity", *IEEE Transactions on Systems, Man, and Cybernetics*, Part A: Systems and Humans, v28 n4, pp. 527-535 (1998).

Busby, J. S., "The Neglect of Feedback in Engineering Design Organizations", *Design Studies*, v19, pp. 103-117 (1998).

Browning, T. R., "Use of Dependency Structure Matrices for Product Development Cycle Time Reduction", *Proceedings of the Fifth International Conference on Concurrent Engineering: Research and Applications*, Tokyo, Japan (1998).

Calantone, R. J. and C. A. Di Benedetto, "Performance and Time to Market: Accelerating Cycle Time with Overlapping Stages", *IEEE Transactions on Engineering Management*, v47 n2, pp. 232-244 (2000).

Calkins, D. E., W. Su and W. T. Chan, "A Design Rule Based Tool for Automobile Systems Design", *Society of Automotive Engineers Technical Paper Series*, Paper 980397 (1998).

Chi, M. T. H., "Quantifying Qualitative Analyses of Verbal Data: A Practical Guide", *The Journal of the Learning Sciences,* v6 n3, pp. 271-315 (1997).

Chimka, J. R. and C. J. Atman, "Graphical Representations of Engineering Design Behavior", *1998 Frontiers in Education Conference,* Session T2D, Tempe, Arizona, November 4-7 (1998).

Christiaans, H. C. M., and K. H. Dorst, "Cognitive Models in Industrial Design Engineering: A Protocol Study", *1992 ASME Design Theory and Methodology Conference,* DE-v42, pp. 131-137 (1992).

Cohen, M., J. Eliashberg and T-H. Ho, "New Product Design Strategy Analysis: A Modeling Framework," in Management of Design: Engineering and Management Perspectives, Dasu, S. and Eastman, C. (Eds.), Kluwer Academic Publishers, pp. 45-60 (1994).

Cooper, K. G., "The Rework Cycle: Part 1: Why Projects are Mismanaged", *Engineering Management Review,* v21 n3, pp. 4-12 (1993).

Cronbach, L. J., "Five Perspectives on Validity Argument", in H. Wainer & H. I. Braun (Eds.), Test Validity (pp. 3-17). Hillsdale, NJ: Lawrence Erlbaum (1988).

Cross, N., Engineering Design Methods. John Wiley & Sons Ltd, New York (1989).

Curtis, B., "Models of Iteration in Software Development", *Iteration in the Software Process: Third International Software Process Workshop,* IEEE (1986).

Delaney, B., "Faster, Better, Cheaper -- NASA Visualizes the Solar System", *IEEE Computer Graphics and Applications,* v17 n6, pp. 10-15 (1997).

Dieter, G. E. Engineering Design: A Materials and Processing Approach, 2nd ed. New York: McGraw-Hill (1991).

Dixon, J. R., "On Research Methodology Towards a Scientific Theory of Engineering Design", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* v1 n3, pp. 145-157 (1987).

Duffy, V. and G. Salvendy, "Relating Company Performance to Staff Perceptions: the Impact of Concurrent Engineering on time to Market," *International Journal of Production Research,* v37 n4, pp. 821-834 (1999).

Dwarakanth, S. and K. M. Wallace, "Decision-making in Engineering Design: Observations from Design Experiments", *Journal of Engineering Design,* v6 n5, pp. 191-206 (1995).

Dwarakanth, S., L. Blessing, and K. Wallace, "Descriptive Studies: A Starting Point for Research in Engineering Design", in Advances in Mechanical Engineering, T. S. Mruthyunjaya (ed.), Narosa Publishing House: New Delhi (1996).

Eide, A. R. , R. D. Jenison, L. H. Mashaw, and L. L. Northup, Introduction to Engineering Design. Boston, MA: McGraw-Hill (1998).

Eisenhardt, K. M. and B. N. Tabrizi, "Accelerating Adaptive Processes: Product Innovation in the Global Computer Industry", *Administrative Science Quarterly,* v40 pp. 84-110 (1995).

Eppinger, S., "Model-based Approaches to Managing Concurrent Engineering". *Journal of Engineering Design,* v2 n4 (1991).

Eppinger, S. D., M. V. Nukala and D. E. Whitney, "Generalised Models of Design Iteration Using Signal Flow Graphs", *Research in Engineering Design,* v9 pp. 112-123 (1997).

Ericsson, K. A. and H. A. Simon, Protocol Analysis : Verbal Reports as Data. MIT Press, Cambridge MA (1993).

Evbuomwan, N. F. O., S. Sivaloganathan, and A. Jebb, "A Survey of Design Philosophies, Models, Methods, and Systems", *Proceedings of the Institution of Mechanical Engineers,* v210, pp. 301-320 (1996).

Finger, S. and J. R. Dixon, "A Review of Research in Mechanical Engineering Design. Part I: Descriptive, Prescriptive, and Computer-Based Models of Design Processes", *Research in Engineering Design,* v1 pp. 51-67 (1989).

Ford, D. N. and J. D. Sterman, "Dynamic Modeling of Product Development Processes", *System Dynamics Review,* v14 n1, pp. 31-68 (1998).

Fricke, G., "Successful Individual Approaches in Engineering Design", *Research in Engineering Design,* v8 pp. 151-165 (1996).

Gebala, D. A. and S. D. Eppinger, "Methods for Analyzing Design Procedures", *Third International ASME Conference on Design Theory and Methodology,* Miami, FL (1991).

Gero, J. S. and T. McNeill, "An approach to the analysis of design protocols", *Design Studies*, v19, pp. 21-61 (1998).

Goel, V., "A comparison of design and nondesign problem spaces", *Artificial Intelligence in Engineering*, v9 pp. 53-72 (1994).

Goldberg, M., "Trendspotting in the New Economy", *The Industry Standard*, Nov 10 (2000).

Gotlieb, L., "Quality Comes to the Information Systems Function", *CMA Magazine*, v66 n7 p. 15 (1992).

Gunther, J. and K. Ehrlenspiel, "Comparing Designers from Practice and Designers with Systematic Design Education", *Design Studies*, v20 pp. 439-451 (1999).

King, E., "Virtual prototyping keys American super car", *Scientific Computing & Automation*, n4, pp. 52-54 (1998).

Konda, S., I. Monarch, P. Sargent and E. Subrahmanian, "Shared Memory in Design: A Unifying Theme for Research and Practice", *Research in Engineering Design*, v4 pp. 23-42 (1992).

Kramlich, J. and J. Fridley, *ENGR Restructuring Team Final Report, Appendix C: ENGR 100 Review/Revision Subcommittee Report*, University of Washington College of Engineering, June 1998. Located at http://www.engr.washington.edu/restruct/engr/appenc.html (1998).

Kusiak, A., J. Wang, D. W. He and C-X. Feng, "A Structured Approach for Analysis of Design Processes", *IEEE Transactions on Components, Packaging, and Manufacturing Technology - Part A*, v18 n3, pp. 664-673 (1995).

Kusiak, A. and N. Larson, "Decomposition and Representation Methods in Mechanical Design", *Transactions of the ASME*, v117 pp. 17-24 (1995).

Legendre, P., "Program *K-means* User's Guide", Departement de sciences biologiques, Universite de Montreal (2001).

Legendre, P. and L. Legendre, Numerical Ecology, 2nd English edition, Elsevier Science BV, Amsterdam (1998).

Love, T., "Constructing a Coherent Cross-Disciplinary Body of Theory about Designing and Designs: Some Philosophical Issues", *Design Studies*, v23 pp. 345-361 (2002).

Madanshetty, S. I., "Cognitive Basis for Conceptual Design", *Research in Engineering Design*, v7 pp. 232-240 (1995).

Malhotra, A., J. C. Thomas, J. M. Carroll and L. A. Miller, "Cognitive Processes in Design", *International Journal of Man-Machine Studies*, v12 pp. 119-140 (1980).

Marples, D. L. "The Decisions of Engineering Design", *IRE Transactions on Engineering Management*, EM-8, pp. 55-71 (1961).

Merriam-Webster Inc., Webster's Ninth New Collegiate Dictionary, Merriam-Webster Inc., Springfield MA (1987).

Milligan, G. W. and M. C. Cooper, "An Examination of Procedures for Determining the number of Clusters in a Data Set", *Psychometrika*, v50 pp. 159-179 (1985).

National Science Foundation, "Research Opportunities in Engineering Design", *NSF Strategic Planning Workshop Final Report*, April 1996.

National Engineering Education Delivery System (NEEDS), Premier Courseware of 2000 (compact disc), University of California (Berkeley), located at http://www.needs.org/engineering/premier/2000/winners.html (2000).

National Engineering Education Delivery System (NEEDS), Premier Courseware of 2001 - (compact disc), University of California (Berkeley), located at http://www.needs.org/engineering/premier/2001/winners.html (2001).

Nukala, M. V., S. D. Eppinger and D. E. Whitney, "Generalized Models of Design Iteration Using Signal Flow Graphs", *1995 ASME Design Theory and Methodology Conference*, Boston MA (1995).

Olson, G. M., J. S. Olson, M. Storrosten, M. Carter, J. Herbsleb and H. Rueter, "The Structure of Activity During Design Meetings", in Design Rationale: Concepts, Techniques, and Use, Lawrence Earlbaum Associates: Mahwah, NJ (1996).

Osborne, S. M., "Product Development Cycle Time Characterization Through Modeling of Process Iteration", S.M. Thesis, M.I.T. Sloan School of Management (1993).

Oxman, R., "Observing the observers: research issues in analysing design activity", *Design Studies*, v16 pp. 275-283 (1995).

Pahl, G. and Beitz, W., Engineering Design: A Systematic Approach. Design Council, London (1988).

Palmer, S.E., "Fundamental aspects of cognitive representation", in Cognition and Categorization, E. Rosch, B.B. Lloyd (eds.), Lawrence Erlbaum Associates, Hillsdale NJ (1978).

Ressler, S. J., "The West Point Bridge Designer", 2000 Premier Award Submission Package, Department of Civil and Mechanical Engineering, United States Military Academy (2000).

Riley, M., "The Beer Recipator 2.2", located at http://hbd.org/cgi-bin/recipator/recipator (1998).

Ringstad, P. R., "Early Component Design Controlled By Decisive Properties", *Journal of Engineering Design,* v7 n1, pp. 39-54 (1996).

Rosenthal, R. and R. L. Rosnow, Essentials of Behavioral Research: Methods and Data Analysis. McGraw-Hill, New York (1984).

Safoutin, M. J., "A Cognitive Model and FMEA Technique for the Analysis of Interdisciplinary Design Teams in Terms of Member Communication", University of Illinois Master's Thesis (1990).

Safoutin, M. J. and D. L. Thurston, "A Communications-Based Technique for Interdisciplinary Design Team Management", *IEEE Transactions on Engineering Management,* v40 n4, pp. 360-372 (1993).

Safoutin, M. J. and R. P. Smith, "Classification of Iteration in Engineering Design Processes", DTM-98-58, *Tenth International ASME Design Theory and Methodology Conference*, Atlanta GA (1998).

Safoutin, M. J., C. J. Atman, R. Adams, T. Rutar, J. Kramlich and J. Fridley, "A Design Attribute Framework for Course Planning and Learning Assessment", *IEEE Transactions on Education,* v42 n2, pp 188-199 (2000).

Schon, D. A., "Problems, Frames, and Perspectives on Designing", *Design Studies*, v9 n3, pp. 132-136 (1984).

Shah, J. J., D. K. Jeon, S. D. Urban, P. Bliznakov and M. Rogers, "Database Infrastructure for Supporting Engineering Design Histories", *Computer-Aided Design,*v28 n5, pp. 347-360 (1996).

Simon, H. A., The Sciences of the Artificial. The MIT Press (1969).

Smith, G. F. and G. J. Browne, "Conceptual Foundations of Design Problem Solving", *IEEE Transactions on Systems, Man, and Cybernetics*, v23 n5, pp. 1209-1219 (1993).

Smith, R. P., S. D. Eppinger and A. Gopal, "Testing an Engineering Design Iteration Model in an Experimental Setting", *1992 ASME Design Theory and Methodology Conference*, DE-v42, pp. 141-147 (1992).

Smith, R. P. and S. D. Eppinger, "Characteristics and Models of Iteration in Engineering Design", *1993 International Conference on Engineering Design (ICED 93)*, pp. 564-571 (1993).

Smith, R. P., "Managing Risk by Reordering Tasks in Engineering Design", *Seventh International ASME Design Theory and Methodology Conference*, Boston, pp. 585-591 (1995).

Smith, R. P., and S. D. Eppinger, "A Predictive Model of Sequential Iteration in Engineering Design", *Management Science*, v43 n 8, pp. 1104-1120 (1997).

Sobek II, D. K., "Understanding the Importance of Intermediate Representations in Engineering Problem-Solving", Working Paper, Mechanical and Industrial Engineering Department, Montana State University, Bozeman MT (2001).

Stauffer, L. A., D. G. Ullman and T. G. Dietterich, "Protocol Analysis of Mechanical Engineering Design", *Proceedings of the International Conference on Engineering Design (ICED 87)*, pp. 74-85 (1987).

Stauffer, L. A. and D. G. Ullman, "Fundamental Processes of Mechanical Designers Based on Empirical Data", *Journal of Engineering Design*, v2 n2, pp. 113-125 (1991).

Stauffer, L. A., M. Diteman and R. Hyde, "Eliciting and Analysing Subjective Data about Engineering Design", *Journal of Engineering Design*, v2 n4, pp. 351-366 (1991).

Stempfle, J. and P. Badke-Schaub, "Thinking in Design Teams - An Analysis of Team Communication", *Design Studies*, in press (2002).

Steward, D. V., "The Design Structure System: A Method for Managing the Design of Complex Systems", *IEEE Transactions on Engineering Management*, v28 n3, pp. 71-74 (1981).

Stewart, D. and D. Hallenbeck, "Three case histories of virtual prototypes to support concurrent engineering", *Professional Program Proceedings, Electronics Industries Forum of New England*, pp. 85-96 (1997).

Stilian, G. N., "PERT: A new management planning and control technique", *AMA Management Report no. 74*, New York: American Management Association (1962).

Suh, N. P., The Principles of Design. Oxford University Press (1990).

Sullivan, W. G., P-M. Lee, J. T. Luxhoj, and R. V. Thannirpalli, "A Survey of Engineering Design Literature: Methodology, Education, Economics, and Management Aspects", *The Engineering Economist*, v40 n1, pp. 7-40 (1994).

Taguchi, G., Introduction to Quality Engineering: Designing Quality into Products and Processes, Asian Productivity Organization (1986).

Terwiesch, C. and C. H. Loch, "Measuring the Effectiveness of Overlapping Development Activities", *Management Science*, v45 n4, pp. 455-465 (1999).

Thilmany, J., "Printing in three dimensions", *Mechanical Engineering*, v123 n5, (2001).

Thomke, S. H., "Managing Experimentation in the Design of New Products", *Management Science*, v44 n6, pp. 743-762 (1998).

Thomke, S. H., E. A. von Hippel, and R. R. Franke, "Modes of Experimentation: An Innovation Process - and Competitive - Variable", *Research Policy*, v27 pp. 315-332 (1998).

Thro, E., The Artificial Intelligence Dictionary, Microtrend Books: San Marcos CA (1991).

Tully, C. J., "Software Process Models and Iteration", *Iteration in the Software Process: Third International Software Process Workshop*, IEEE (1986).

Ullman, D. G., T. G. Dietterich and L. A. Stauffer, "A Model of the Mechanical Design Process Based on Empirical Data", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, v2 n1, pp. 33-52 (1988).

Ullman, D. G., "A Taxonomy of Mechanical Design", *ASME Design Theory and Methodology Conference - DTM '89*, Montreal, pp. 23-36 (1989).

Urban, G. L. and Hauser, J. R., Design and Marketing of New Products, 2nd edition, Prentice-Hall: Englewood Cliffs NJ (1993).

Valkenburg, R. and K. Dorst, "The reflective practice of design teams", *Design Studies,* v19 pp. 249-271 (1998).

Wagoner, R., "Create a Real Time Company", in article "How to Succeed in 2003", *Business 2.0*, v3 n12, pp. 87-98 (2002).

Wallace, K. M. and C. Hales, "Detailed Analysis of an Engineering Design Project", *Proceedings of the International Conference on Engineering Design (ICED 87),* pp. 94-101 (1987).

Wileden, J. C., "Incremental Development and Iteration in the Software Process", *Iteration in the Software Process: Third International Software Process Workshop,* IEEE (1986).

Williams, T., C. Eden, F. Ackermann and A. Tait, "Vicious circles of parallelism", *International Journal of Project Management*, v13 n3, pp. 151-155 (1995).

Zar, J. H., Biostatistical Analysis. Third Edition, Prentice-Hall: Englewood Cliffs NJ (1996).

Appendix A

Essay Questions and Homework Assignment

Essay questions

As part of the homework assignment through which data was collected, each subject was asked to answer several essay questions relating to their experience in solving the problem. These questions are described below.

Question 1: Design goal / Preconception

In Fall 2001, the first essay question asked whether the student had focused primarily on distance, primarily on speed, or a combination of both:

> When you began designing, did you already know the type of car that you would design, or did you design the car as you worked with the software?

In Winter 2002, students were asked about whether the student had begun using the software with a specific design in mind, or had evolved the design entirely during the session:

> What were your objectives as you designed the car? For example, did you focus mainly on speed, mainly on distance, or a combination? Did you have any other objectives?

Question 2: Description of process

The next question asked for a verbal description of the process by which the design evolved. Suggested topics included changes in focus from one objective to another, difficulties encountered with each objective, any sort of design strategy that was followed, and how the subject arrived at the decision to halt the process:

> Describe the steps you followed as you developed your design. For example, maybe you first focused on one component of the car, and then shifted to another. Which ones, and how? Did you evolve some sort of design strategy as time went by? How did you know when to stop?

Question 3: Meeting of objectives

Subjects were then asked to comment on how well their final car design meets their original objectives, and why they felt so:

> Do you feel that the final car design meets your design objectives? Why or why not?

Question 4: Self-rating of iterativeness

Subjects were asked to provide a self-rating of the iterative character of their process using a scale in which 1 represents a linear or noniterative process and 10 represents a highly iterative process.

> People approach design problems in different ways. Some people tend to design in a linear fashion. They try to make very careful decisions up front, and do relatively little testing and modification. Others design in an iterative fashion. They spend less time on decisions up front, and more time trying out many different things, repeating the cycle over and over until things are just right. Both methods can be very effective. Which fashion do you think describes your design process the best? Give examples of why you think so. If you had to rate your design process where 0 is "very linear" and 10 is "very iterative", what number would you give it?

Question 5: Familiarity with software

Subjects were asked about their degree of familiarity with the software prior to starting the design process for the homework assignment, and whether or not they felt that this familiarity or lack of it affected their design process.

> How easy was it to use the Virtual Car software? Did it become easier to use over time, or were you already familiar with it? Do you think it affected your design process?

Homework Assignment

The following pages show these questions in the context of the actual homework assignment as it was assigned to the subjects in Winter 2002.

Virtual Car Homework #2:
Your Personal Design

As a member of a design team, you have a responsibility to bring your best ideas to the table. Four heads are better than one where creativity is concerned! To get your team started, use Virtual Car Version 4 to design a car as if you were going to enter it into the Speed Competition. Later, we will run a Virtual Race, where the car design that you turn in will be loaded into Virtual Car and raced against everyone else's design.

Experiment with the shape of the car, the dimensions of the spring, the wheel diameters, and so on, until you have designed a single car that you think would do pretty well in the Speed Competition.

- Your car has to have a projected distance of at least 15 feet to qualify.
- Your car should have no skid advisory or any other kind of advisory, i.e. the Advisories window should not be popping up.

Directions

1. For this assignment you must use Virtual Car Version 4.03a. Follow these directions to get your copy and to create your Design Portfolio file.
2. When you first start Virtual Car, create a single design portfolio file, and work with only that portfolio until your design is done.
3. You may complete the design in a single session, or continue designing it over several sessions. Between sessions, or if you change computers, be sure to keep all of the .vcdp and .vcar files with you, or you will have to start over.
4. When your design is completely done, use the Print button to print a copy of the Summary Sheet.
5. Finally, copy your design portfolio file (ends with .vcdp) and the file that contains your final design (ends with .vcar) to a floppy disk or to Dante, so you can turn them in.

Deliverables

When finished, please turn in the following items:

1. Your design portfolio file (such as MyDesignPortfolio.vcdp) by floppy disk or email.
2. The car design file in which you saved your final design (such as MyDesign.vcar) by floppy disk or email.
3. A printed copy of the Report sheet (use Virtual Car's Print button, and select Print Report Only).
4. Your answers to the following discussion questions, in memo format:

Questions

1. When you began designing, did you already know the type of car that you would design, or did you design the car as you worked with the software?

2. Describe the steps you followed as you developed your design. For example, maybe you first focused on one component of the car, and then shifted to another. Which ones, and how? Did you evolve some sort of design strategy as time went by? How did you know when to stop?

(continued)

3. Do you feel that the final car design meets your design objectives? Why or why not?

4. People approach design problems in different ways. Some people tend to design in a linear fashion. They try to make very careful decisions up front, and do relatively little testing and modification. Others design in an iterative fashion. They spend less time on decisions up front, and more time trying out many different things, repeating the cycle over and over until things are just right. Both methods can be very effective. Which fashion do you think describes your design process the best? Give examples of why you think so. If you had to rate your design process where 0 is "very linear" and 10 is "very iterative", what number would you give it?

5. How easy was it to use the Virtual Car software? Did it become easier to use over time, or were you already familiar with it? Do you think it affected your design process?

GRADING **Weight: 7.5% of course grade**	
ITEM	**PERCENT**
Time to 15 ft in Virtual Race	30
Essay questions	40
Design Portfolio (.vcdp) file turned in	15
Car Design (.vcar) file turned in	15
TOTAL	**100**

Appendix B

University of Washington Consent Form

UNIVERSITY OF WASHINGTON CONSENT FORM
"Identifying Iterative Design Behavior for Design Process Representation"

Michael J. Safoutin, PhD Candidate, Industrial Engineering, 616-9828
Cynthia J. Atman, Faculty Sponsor, Industrial Engineering, 616-2171

Investigator's Statement

We are asking you to contribute data to a research study. The purpose of this consent form is to give you the information you will need to help you decide whether or not to contribute data to the study. Please read the form carefully. You may ask questions about the purpose of the research, what we would ask you to do, the possible risks and benefits, your rights as a volunteer, and anything else about the research or this form that is not clear. When all your questions have been answered, you can decide if you want to contribute data or not. This process is called 'informed consent'.

PURPOSE AND BENEFITS

We want to develop methods to represent differences among design processes that are performed by different designers while they pursue the same design problem. This research could lead to ways to tell the difference between effective design strategies and those that are less effective. We hope the results of this study will help us improve design management and the way we evaluate design education. You may not directly benefit from this research.

PROCEDURES

If you choose to take part in this study, we would like to use components of the Virtual Car project. We would only like to use some of your work related to this design project. We will not ask you to provide any more work than is regularly assigned. We would like to analyze the following: (a) the file that you turned in for the individual design homework, in which you designed your first car using Virtual Car; (b) the essay homework that was collected with this assignment, and (c) the file that results from the use of Virtual Car by your group during the project. We will only use group projects for which all members have consented to be in the research. We will replace names with randomly assigned codes on all of the assignments and projects before they are analyzed for this research. The random number will not be linked to your name, or the group's identity.

RISKS, STRESS, OR DISCOMFORT

Some people may feel uncomfortable having their class work included in research, even if it is confidential.

OTHER INFORMATION

Participation in this study is voluntary. Your assignments and projects are initially linked to your name. However, information that would identify you (like your name or group identification) is replaced with a code before the study information is analyzed. You can change your mind about taking part in this study before the assignments are coded. Your instructor will not know if you took part in this study until after the course is over and grades have been assigned. Your decision will not affect your grade or your standing in the course in any way. If we publish the results of this study, we will not use your name.

_____	_____	_____
Signature of investigator	Printed name	Date

SUBJECT'S STATEMENT

___ I wish to participate. The study has been explained to me and I have had an opportunity to ask questions. I give permission for the instructor to extract data from the homeworks and activities described above and to use the data in research. I understand that future questions I may have about the research or about my rights as a subject will be answered by one of the investigators listed above.

___ I do not wish to participate.

_____	_____	_____
Signature of subject	Printed name	Date

Version of November 12, 2001

Appendix C

Timeline Coding Notes

Timeline Coding Notes

Prior to analysis, the printed timelines were inspected to identify any interpretive difficulties and develop a systematic procedure for analysis. Some minor uncertainties regarding the proper interpretation of plotted symbols and sequences of events were identified and resolved. Some were traced to minor programming oversights that occasionally changed the appearance of plotted events. Other uncertainties related to choosing an appropriate interpretation for certain situations. These uncertainties and their resolution are described below.

- The last feedback event in a process was not evaluated for parameter identity because by definition no parameters can be carried over to the next event.

- Multiple edits to the same parameter were counted as a single edit if no feedback was supplied between each edit.

- A circular symbol representing the application of a parameter change could occasionally appear out of place. Rather than depicting a single symbol to represent a single assimilation event, several types of symbols were used to indicate components of a single event: a narrow rectangle was used to indicate each individual keystroke as the user typed several characters of a parameter value; a wider rectangle indicated that the design parameter had been exited (i.e. the typing was finished if keystrokes were involved), and a circle indicated that the parameter setting had been explicitly "applied" to the design. The "apply" event was later was found to have little informative value but was retained in the software. The circle corresponding to the "apply" event could be generated in three ways: by the designer explicitly pressing a dummy button marked "Apply", pressing the Return key after typing the parameter value, or activating a feedback mode. Due to a programming oversight, the circle could also be erroneously generated in certain circumstances after the designer loaded a car design from disk or switched from one default design to another. These circles are erroneously issued to enforce an "apply" event for parameters that were "edited" via the loading of the saved or

default car design. Because the Apply event was not relevant in for identifying assimilation and feedback, these erroneous circles were merely ignored.

- Depending on whether the user tended to use the Tab key or a mouseclick to switch between parameters, and on whether the Return key or the Apply button was pressed, it was possible for some symbols to vary slightly in their appearance by having the square symbol that represents "exit" be missing. This error is of no importance because the square is not necessary to mark an assimilation event.

- Although loading a new car design could be interpreted as the editing of parameter values (i.e., the entire set of 24 parameters), they were disregarded because they do not represent specific parameter choices made by the designer. The true purpose of the timeline in characterizing a design process is not in documenting aggregate or net changes to parameter values over time, but in representing localized patterns in the decision path of the designer with respect to intentional parameter visitations and requests for feedback.

- A general decision was made to count a parameter edit as valid if either of the following were true: (a) the designer explicitly performed the parameter change by editing the parameter directly, or (b) the parameter change was represented in a feedback event. This means that several potential situations were not counted as parameter changes: (a) when the designer chose one parameter setting but changed it back to its original value before getting any feedback, or (b) an event was automatically fired as an outcome of a different parameter setting, but the parameter setting that caused it to be changed is itself changed back before feedback is received (which incidentally causes the other parameter to revert as well). For example, when selecting a four-wheel-drive configuration for a car that was previously rear-wheel drive, it is usually necessary to automatically increase the size of the front wheel because the software requires that both wheels be the same size. This automatically generates an event that indicates a change to the front wheel size parameter. If the designer switches back to 2WD, the front wheel registers another automatic edit event as it is automatically returns to its original 2WD-size. In a few cases, however, the designer immediately switched back to a rear wheel drive configuration without seeking feedback about the 4WD configuration. In these

cases the automatically fired events signifying both changes to the front wheel size were disregarded because these changes were not explicitly requested by the designer and were never reflected in a feedback result.

Appendix D

Iterativity Ratio

Iterativity Ratio: A Measure of Iteration Based on the A-F Timeline

An *A-F* timeline depicts individual feedback and assimilation events and episodes. This allows them to be discerned and tallied over the course of the depicted process. In this and subsequent discussion, the total number of assimilation events in a process will be referred to as E_a, and the total number of feedback events will be referred to as E_f. The number of assimilation episodes and feedback episodes in a process will be denoted by S_a and S_f respectively.

Iterativity Ratio

The *A-F* timeline suggests that a design process might be characterized in terms of the relative degree to which it generates feedback as opposed to assimilating information from the design environment. Processes that are feedback iterative generate feedback information; it stands to reason, then, that a process that is particularly iterative in this manner would be expected to proportionately generate feedback. By contrast, a process that is less feedback iterative would be dominated by assimilation of information from the design environment rather than from feedback, and would be indicated by a relatively lower proportion of feedback events.

Imagining a process that is maximally iterative in this regard, every modification to the design (that is, every assimilation event) would be immediately followed by a feedback episode. That is, no single parameter setting escapes direct evaluation via feedback. Under an appropriate metric, which we will refer to as *I*, such a process would evaluate to unity. Two examples of such a process are depicted in Figure D.1. In Figure D.1(a), assimilation and feedback events simply alternate, indicating that every parameter setting is followed by a feedback event. Figure D.1(b) depicts essentially the same situation in which every parameter setting is followed by a feedback episode of one or more feedback events. The mark of a fully iterative process in this regard is in the fact that every assimilation *event* is evaluated by *one or more* feedback events.

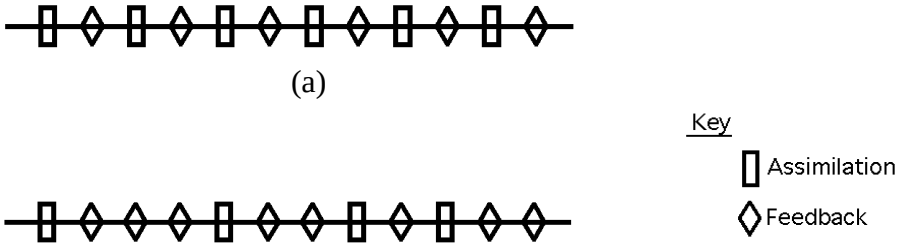


Figure D.1. Processes with I = 1

By contrast, in a minimally iterative process, no assimilation event is ever evaluated by feedback. An example of this process is depicted in Figure D.2. Under an appropriate metric *I* this process would evaluate to zero.

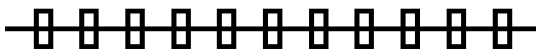


Figure D.2. Process with I = 0

Processes evaluated under this metric would thus approach *I = 1* to the degree that parameter settings tend to be evaluated directly before the next setting is made, or would approach *I = 0* to the degree that parameter settings are made without prior feedback.

A metric that accomplishes this task is simply the ratio of feedback *episodes* to assimilation *events*. This metric will be referred to as *iterativity*, denoted by the label ***I***:

Iterativity = number of feedback episodes / number of assimilation events

$$I = S_f / E_a$$

An iterativity ratio may either be computed for an entire process to characterize its average emphasis on feedback, or to segments of the process to provide "local" iterativity measures at various points in the process. The iterativity ratio carries interesting

implications that suggest utility of the measure in design research. In a design situation in which the cost of obtaining feedback is relatively high, one would expect a relatively low iterativity value because a natural reaction would be to submit more parameter changes per feedback request. On the other hand, in a design environment in which synthesis of design decisions is difficult, costly, or unreliable, one might expect the designer to rely more heavily on analysis through feedback, causing the process to have a relatively high iterativity value. In an experimental setting, one might explore this relationship by manipulating these variables, observing their effect on $\boldsymbol{I}$, and relating I to design outcome.

Appendix E

Computed Version of Feedback Quality Measure

A Computed Version of the Feedback Quality Measure

The feedback quality measure may alternatively be applied in a computed rather than discrete manner, resulting in a single point on a two dimensional continuum rather than a discrete categorization. This not only accounts for parameter quantity more accurately, but also allows individual processes to be readily compared to one another on a single plot.

Each feedback event is given a parameter quantity score and a parameter identity score. Parameter quantity is computed as the inverse of the number of parameter changes active in each feedback event, averaged over all events. Parameter identity is computed as a probability that a parameter edited prior to one feedback event will also be active in the next event. This is computed by counting the total number of times any parameter edit is continued across a feedback event and dividing by the total number of parameter edits.

The computed application results in a parameter quantity score and a parameter identity score that may be combined into an ordered pair and plotted on a coordinate plane bounded by (0,0) and (1,1) as depicted in Figure E.1:

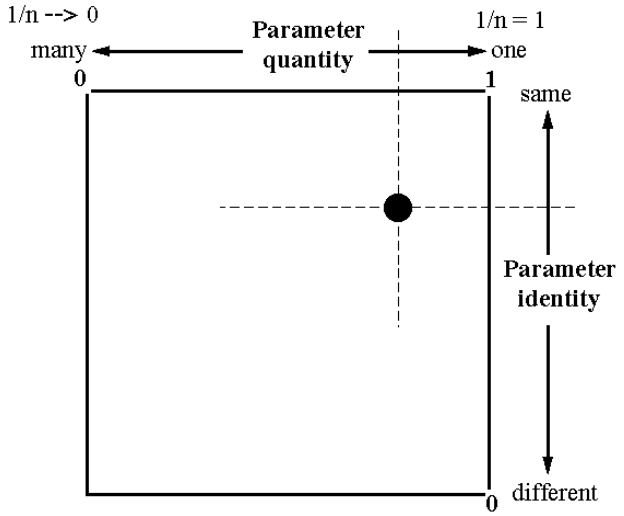


Figure E.1. Example plot of computed feedback quality

Comparing the Discrete and Computed Applications

It is important to note that one should not expect a direct dimensional correlation between the discrete and computed depictions for a given process. The discrete application is a categorization of individual processes within a dimensionless quadrant system, while the computed application assigns it a value in a continuous numeric space. For example, a process that is predominantly Class A under the discrete categorization would not necessarily have a computed value in the upper right quadrant of the continuous plot. The continuous plot is not a quadrant space; its extents are defined differently and do not share the same extremes.

Furthermore, while the discrete application assesses parameter quantity by simply classifying each feedback event as confounded ($n > 1$) or not confounded ($n = 1$), the computed application weights each event according to the actual number of confounding variables n. Because of this, the computed value could be shifted below or to the left of the *apparent* "A" quadrant if a minority of events are extremely confounded Type B or Type C. For the same reasons, individual processes that are very similar under the discrete A-D characterization could differ significantly when expressed under the computed application. Generally, however, processes that are predominantly Class A should tend toward the upper right area of a group of plotted processes.

Analysis of Computed Feedback Quality

Table E-1 reports computed values for parameter quantity and parameter identity by subject. These values are placed into scatter plots in Figures E.2 and E.3. Points nearest the upper-right corner of the plots are associated with the highest quality feedback because in this region parameter continuity is maximized and parameter confounding is minimized. The space depicted in this plot is not a quadrant space and should not be confused with the 2x2 grid of the discrete feedback quality representation.

Table E-1. Computed parameter quantity and parameter identity

	Fall 2001	
	Quantity	Identity
K235	0.90	0.12
K238	0.94	0.60
K333	0.74	0.21
A411	0.92	0.54
A433	0.88	0.42
B241	0.85	0.45
C695	0.61	0.34
D477	0.79	0.41
W255	0.97	0.56
W455	0.71	0.12
X250	0.81	0.22
X397	0.84	0.28
Y318	0.85	0.39
Y389	0.85	0.60
Y402	0.88	0.40
Y418	0.85	0.36
Mean	**0.84**	**0.38**
StDev	**0.09**	**0.16**
Min	**0.09**	**0.12**
Max	**0.97**	**0.60**

	Winter 2002	
Subject	Quantity	Identity
A185	0.46	0.88
A368	0.32	0.94
A446	0.46	0.92
A527	0.46	0.93
A742	0.28	0.77
F234	0.23	0.74
F265	0.45	0.88
F336	0.95	0.56
R246	0.85	0.42
R815	0.90	0.45
R819	0.41	0.48
L258	0.86	0.46
L564	0.92	0.64
U487	0.86	0.40
U750	0.89	0.46
U767	0.95	0.51
G357	0.80	0.27
G365	0.91	0.43
G654	0.76	0.52
E150	0.93	0.55
E747	0.82	0.16
H45	0.85	0.32
H456	0.89	0.43
H837	0.55	0.34
Mean	**0.70**	**0.56**
StDev	**0.25**	**0.23**
Min	**0.23**	**0.16**
Max	**0.95**	**0.94**

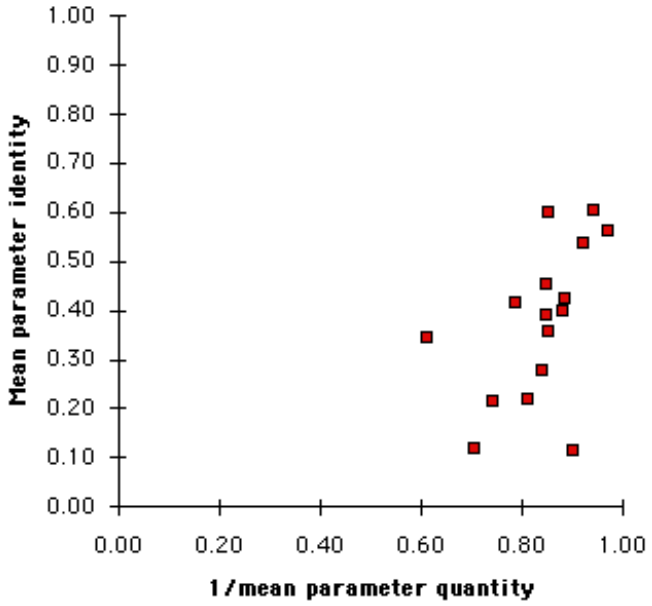


Figure E.2. Computed feedback quality, Fall 2001

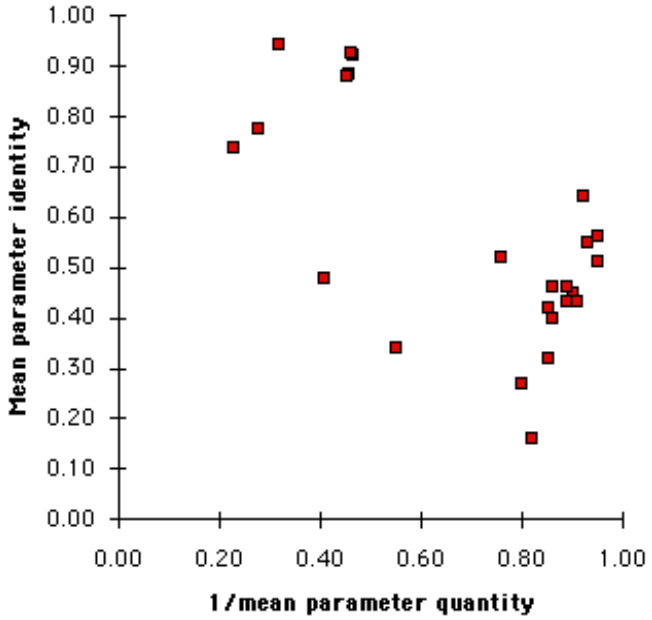


Figure E.3. Computed feedback quality, Winter 2002

Comparison: Computed Feedback Quality

The two data sets exhibit quite different mean values for parameter quantity and parameter identity. The Fall processes on the average score better in parameter quantity (0.84 vs. 0.70) but lower in parameter identity (0.38 vs. 0.56). The Fall data also exhibits somewhat tighter standard deviations in both variables (0.09 and 0.16 vs. 0.25 and 0.23).

The relative clustering of points on the scatter plots provide additional evidence of these differences. While the Fall data is quite well confined to one major cluster extending upward to the right, the Winter data consists of two distinct clusters and several outliers. The largest cluster resembles a somewhat tighter version of the Fall cluster, but the secondary cluster is in a region defined by high parameter identity and low parameter quantity.

Clustering Methods

The next task is that of grouping processes based on their similarity in terms of the computed measure. Although this might be done visually by simply looking at the plot, a number of analytical approaches are available.

In cases like this where there is no preexisting classification scheme that is known to apply, and no basis to suggest a canonical classification scheme, it is possible to divide the objects into so-called "natural" categories that are implied by the distribution of the objects themselves. This is known as *cluster analysis.* Cluster analysis is a popular technique for grouping data describing empirical observations, and is used in a broad variety of fields ranging from biological taxonomy, marketing, and gene analysis [Afifi and Clark 1984]. In a clustering problem there are two major questions to be answered if a meaningful grouping is to be found. First, how many clusters should the objects be clustered into, and second, where should each cluster be centered relative to the others? One popular criterion for an ideal clustering requires that each object reside in the cluster whose centroid is nearest the object, and the clusters have been chosen so that the sum, over all groups, of the squared mean distance of objects in a cluster from the centroid of their cluster has been minimized [Legendre 2001]. In other words, the number and

location of clusters has been selected so that, when every object is assigned to its nearest cluster, the resultant clusters are as geometrically compact as possible.

K-means clustering is a popular and well established clustering technique [Legendre and Legendre 1998], [Afifi and Clark 1984]. Many algorithms for k-means clustering have been described, but all of them operate on the same core principle. The algorithm first requires that the number of clusters *k* be specified. This may be specified by the user, but more commonly the algorithm is run in a loop with a range of *k* values to be tried. Next, *k* bins are established and each object is randomly assigned to a bin to form an initial set of clusters. The geometric centroid of each random cluster is then computed (the centroid is simply the mean value of each variable describing the objects in the cluster). Each object in each cluster is then examined to see if it may be reassigned to a different cluster whose centroid is geometrically nearer; otherwise it is left in the same cluster. New centroids are computed, and each object assignment is again examined, in an iterative cycle. After no more objects may be reassigned, the sum of the squares of the within-group residuals (i.e. the objective function) is calculated and saved as a measure of the quality of the clustering.

Unfortunately, partitioning an arbitrary set of objects into groups based on similarity is an NP-hard problem. Because the initial grouping of objects into clusters is random, a single run of the algorithm will not necessarily deliver the optimum partitioning that would minimize the objective function, but may instead converge to a local minimum. Most k-means algorithms include an outer loop that repeats the entire algorithm many times with a different random seeding of clusters. After a large number of repetitions, the clustering result that delivered the lowest value of the objective function is selected as the most likely optimum clustering.

If the algorithm has been run with several different *k* values, the optimum *k* value may be identified by applying the Calinski-Harabasz criterion [Legendre 2001] and selecting the *k* value that maximizes it. This criterion is built into many k-means algorithms and its background is beyond the scope of this work. A survey of this and other stopping rules for cluster analysis may be found in Milligan and Cooper [1985].

A wide variety of statistical tools are available to perform k-means clustering, ranging from computational modules for standard statistical packages such as Matlab and Mathematica, to small standalone programs. A standalone open source statistical program called *K-means* [Legendre 2001] was used in the current study.

Clustering Based on Computed Feedback Quality

Individual processes were next clustered in terms of similarity in their computed feedback quality measures. Clustering was performed by a *k-means* program [LeGendre 2001] using 100 repetitions of initial random seed clustering while looping through a range of *k* values from 12 to 2.

For Fall 2001, the optimum number of clusters *k* was reported to be between 6 and 8. Tables E-2 and E-3 depict the optimal clusters for *k=6* and *k* =8, respectively. For Winter 2002, the optimum number of clusters was reported to be between 7 and 9. Tables E-4 and E-5 depict the optimal clusters for *k*=7 and *k* = 9, respectively.

Table E-2. Optimal k-means grouping of computed feedback quality, k=6, Fall 2001

1	2	3	4	5	6
K238	A433	K235	X250	W455	C695
A411	B241	W255	X397		
Y389	D477				
	Y318				
	Y402				
	Y418				
	K333				

Table E-3. Optimal k-means grouping of computed feedback quality, k=8, Fall 2001

1	2	3	4	5	6	7	8
K238	D477	A433	Y318	K235	K333	W455	NC695
A411		B241	Y402		X250		
W255			Y418		X397		
Y389							

Table E-4. Optimal k-means grouping of computed feedback quality, k=7, Winter 2002

1	2	3	4	5	6	7
F336	R246	A185	G357	A742	R819	H837
L564	R815	A368	E747	F234		
U767	L258	A446	H245			
E150	U487	A527				
	U750	F265				
	G365					
	G654					
	H456					

Table E-5. Optimal k-means grouping of computed feedback quality, k=9, Winter 2002

1	2	3	4	5	6	7	8	9
L564	F336	R246	A185	G357	G654	A742	R819	H837
	U767	R815	A368	E747		F234		
	E150	L258	A446	H245				
		U487	A527					
		U750	F265					
		G365						
		H456						

Because the primary role of a *k-means* algorithm is simply to place items in clusters, the clusters that are returned have no particular ordering. The algorithm only assigns an arbitrary number to each cluster in order to uniquely identify it. For maximum clarity, in the tables, each cluster was placed in a position roughly corresponding to the position that their members occupy in Tables 7-3 and 7-4 of Chapter 7.

The result of clustering the computed feedback quality measures did not perfectly match the clusters that were generated for the discrete feedback quality measures. Overall, however, the computed feedback quality clusters match quite well with the discrete feedback quality clusters. In both the Fall and Winter data sets, a visual inspection reveals that the vast majority of individual processes are either clustered with the same counterparts in both representations, or their former counterparts have move to the next adjacent cluster.

Differences in Computed Feedback Quality Classification

There are at least two reasons to expect minor disparities between a classification based on discrete feedback quality and one based on computed feedback quality . First, the computed application accounts for the actual degree of parameter quantity in each feedback event rather than simply counting it as confounded or not confounded. This would tend to cause highly confounded processes, i.e. those that typically submit many parameters to a feedback event, to place relatively farther left via the computed feedback quality than it did under the discrete feedback quality . Meanwhile, its peers that tend to confound with fewer parameters would experience less of this effect. Second, the computed feedback quality measures have undergone a "natural" clustering rather than being assigned to canonical categories. Neither the number of clusters nor their geometric composition would necessarily be identical to the discrete case, leading to minor variations in cluster membership.

Appendix F

Alternative Definitions of a Random Process

Alternative Definitions of a Random Process

In the analysis of Chapter 7, a random process was assumed to result in an equal likelihood of a feedback episode being classified as A, B, C, or D. This would result in the expected profile shown in Figure F.1.

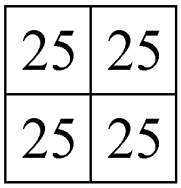


Figure F.1. Expected profile of a random process, base assumption

If a group of random processes of finite length were to be observed, each individual process would vary somewhat from the expected distribution, with an equal likelihood of favoring any one of the quadrants. The distribution of individuals in a randomly designing group would thus appear as in Table F-1:

Table F-1. Expected distribution of a randomly designing group, base assumption

A-dominant	B- and D-dominant	C-dominant
25	50	25

Is this a reasonable assumption? In defining a random design process, the intent is to define a process in which a human designer makes all design decisions by random chance rather than making these decisions with respect to their expected impact on the design goal. That is, the following decisions are made at random: (a) whether the next event should be one of assimilation or of feedback, (b) if assimilation, which parameter should be edited, and (c) if feedback, which functional requirement should be evaluated.

A more detailed examination will reveal that the actual expected result of a random process depends on the number of design parameters that are available for editing, and on

designer awareness of these parameters. Therefore a judgement must be made when selecting a model for a random process. The issues surrounding this judgement are outlined in the following discussion.

A random process has clear implications for the parameter quantity metric. Parameter quantity (that is, the number of parameters submitted per feedback request) determines the relative share between quadrants A/D (single parameter) and quadrants B/C (multiple parameter). Figure F.2 depicts the mechanism by which this is determined in a random process. At any point at which a single parameter has been edited, random chance dictates that (a) there is a 50% chance that the next event will be another assimilation event, and (b) there is a 50% chance that it will be a feedback request.

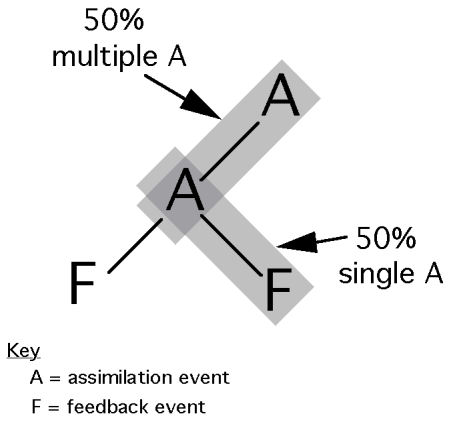


Figure F.2. Effect of random choice on single or multiple parameter quantity

In the former case, the assimilation episode now consists of multiple events, and thus will be rated either as B or C. In the latter, the assimilation episode terminates with a single event and will thus be rated either A or D. Thus a random process would produce 50% single parameter feedback events (A and D) and 50% multiple parameter events (B and C), as depicted in Figure F.3.

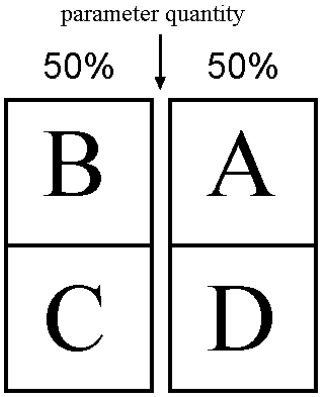


Figure F.3. Expected parameter quantity for a random process

Now let us consider the relative apportionment between the 50% that is shared between B and C and the 50% that is shared between A and D. This is determined by the behavior of the parameter identity metric. Here, the result of a random process will vary depending on the number of parameters that are eligible for editing. The greater the number of parameters, the lower the probability that a parameter in one assimilation episode will be present in the next. Therefore, in a random process, a task with a large number of parameters tends to weight the expected measure toward the lower quadrants (C, D) and away from the upper quadrants (A, B). The expected weighting toward (C, D) may be calculated mathematically based upon the expected number of parameters edited per assimilation episode (*E*) and the number of design parameters available (p), as shown in Figure F.4.

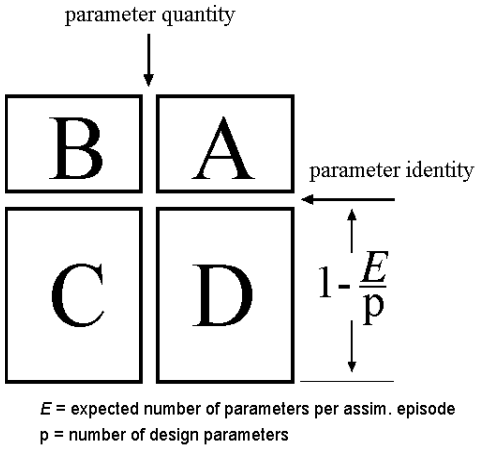


Figure F.4. Expected profile of a random process determined by *E* and p

From this figure it is clear that a design task with a large number of parameters (indicated by a large value for p) will strongly favor Type C and D patterns, while the expected share between (A,D) and (B,C) is equal regardless.

What then is an appropriate model for a random process with respect to the Virtual Car design task? As discussed in Chapter 5, this task has 24 available design parameters. By simulation, it was determined that the expected number of parameter edits per assimilation event is two. This leads to an expected distribution among the quadrants of 4.2% A, 4.2% B, 45.8% C, and 45.8% D, as shown in Figure F.5.

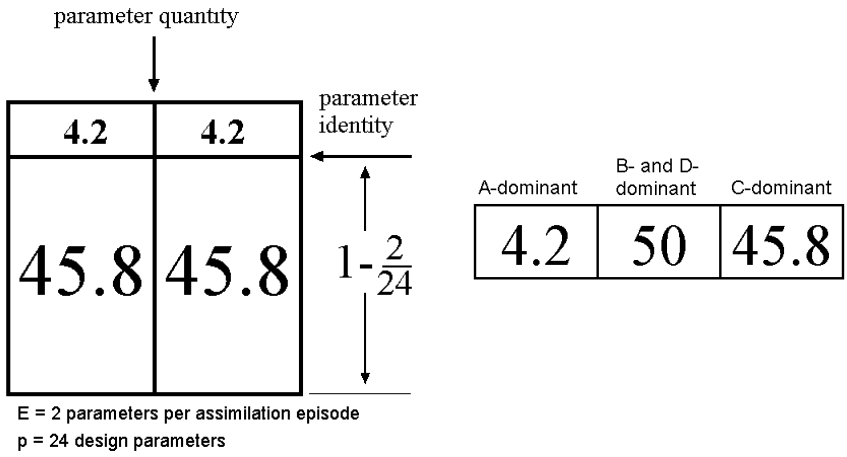


Figure F.5. Expected profile of a random process for E = 2 and p = 24

The expected profile 4.2 A / 50 BD / 45.8 C differs substantially from the base assumption of 25 A / 50 BD / 25 C. As an expected random result, it also differs much more sharply from the observed processes, most of which are very strong in Type A. If this model is employed in the analysis of Expectations 1 and 3 in Chapter 7, the case for both expectations is strengthened due to the sharper variation.

However, to adopt this model for a random process is to assume that the designer is aware of the existence of all 24 design parameters and considers each in randomly choosing which to edit. Judging from the activity of the observed subjects, it may be more reasonable to assume that the designer is not aware of all of the parameters. For example, on the average, the subjects in the Fall and Winter groups edited only 11 to 12

parameters during the course of the process. Because the subjects were relatively new to the design task, it is possible that they were simply unaware of the existence of some of the available parameters. A reasonable reaction would be to reduce the assumed number of parameters to account for designer awareness.

At one extreme, reducing the number of parameters from 24 to 4 reveals (by simulation) an expected distribution that closely resembles our base assumption of equal likelihood, as shown in Figure F.6:

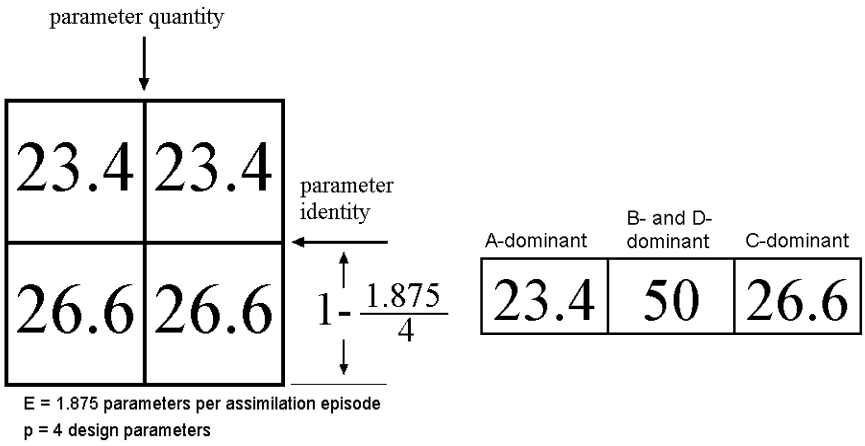


Figure F.6. Expected profile of a random process for E = 1.875 and p = 4

In conclusion, any of these alternative versions of a random process would act to strengthen the case for Expectations 1 and 3 if they were adopted instead of the base assumption employed in Chapter 7. The base assumption employed in Chapter 7 resembles most closely a design task with four parameters. This assumption is a more conservative assumption than those that would attempt to account for the actual number of parameters available in the task that was assigned to the subjects. In particular, the effect observed in the Fall group becomes significant when one adopts an assumption that approximately eight or more design parameters are considered.

Vita

Michael J. Safoutin received his Master of Science from the University of Illinois at Urbana-Champaign in 1990. In 1991 he joined the Director's office of the EPA National Vehicle and Fuel Emissions Laboratory and became involved with energy modeling and policy issues related to hybrid and alternative-fuel vehicles. At the University of Washington he coordinated the design and operation of the Integrated Learning Factory design education facility, performed research for the Center for Engineering Learning and Teaching (CELT), and was named the 2003 Outstanding Teaching Assistant for the College of Engineering in connection with his work with Engineering 100, Introduction to Engineering Design. In August 2003 he received his Doctor of Philosophy in Mechanical Engineering, with specialization in Industrial Engineering. His research interests include design theory and methodology, design management, artificial intelligence in design, transportation energy utilization, energy and the environment, and engineering education.