Tripp, Steven D. & Bichelmeyer, Barbara. (1990) Rapid Prototyping: An Alternative Instructional Design Strategy.

Rapid Prototyping: An Alternative Instructional Design Strategy

Eli Steven D. Tripp and Barbara Bichelmeyer
Steven D. Tripp and Barbara Bichelmeyer are in the Instructional Technology Center, the University of Kansas, Lawrence, Kansas 66045.

There is a design methodology called rapid prototyping which has been used successfully in software engineering. Given the similarities between software design and instructional design, we argue that rapid~ prototyping is a viable model for instructional design, especially for computer-based instruction. Additionally, we argue that recent theories of design offer plausible explanations for the apparent success of rapid prototyping in software design. Such theories also support the notion that rapid prototyping is appropriate for instructional design. We offer guidelines for the use of rapid prototyping and list possible tradeoffs in its application.

The standard rationale for the "systems approach" to instructional design has been the effectiveness of the product rather than the efficiency of the process (Branson & Grow, 1987; Briggs, 1977; Briggs & Wager, 1981; Gagne, 1987; Gagne & Briggs, 1979). It is rarely argued that the systems approach is efficient. Indeed it is sometimes admitted that ISD is costly. For example, Romiszowski (1981) acknowledges this inefficiency by stating that, "... when one is venturing into *instructional design,* which is quite expensive, one should justify the cost" (p. 157). There is always a need for design methodologies which are more efficient, while maintaining or enhancing effectiveness. A software design methodology called *rapid prototyping* has recently been advocated because it solves efficiency problems associated with traditional software design methods while increasing effectiveness. The purpose of this paper is to evaluate why this design methodology may be appropriate to instructional systems design.

THE NATURE OF DESIGN

In 1969, before personal computers became standard equipment in many homes, businesses, and schools, and before great numbers of college students were planning careers in computer technology, Herbert Simon addressed an issue that was to become important to software developers and instructional designers. Simon (1981) spoke to the issue of the nature of fields like computer science, engineering, and education by pro

posing a difference between the natural sciences and what he called *sciences of the artificial.* The disciplines which Simon defined as artificial include (but are not limited to) engineering, medicine, architecture, and instruction. The four qualities that separate the natural sciences from the artificial or design sciences are (a) artificial things are synthesized by people; (b) artificial things imitate appearances of natural things but lack the reality of them; (c) artificial things can be characterized in terms of functions, goals, and adaptation; and (d) artificial things are usually discussed in terms of imperatives as well as descriptives.

Simon's theory that there are important differences between the natural and artificial sciences was empirically substantiated by Lawson (reported in Lawson, 1980), who conducted a study to discover the differences between natural scientists and architects in design-like problem-solving. In Lawson's study, the problem was to arrange colored blocks onto a 3-by-4 rectangular pattern with the objective being to show as much red or blue as possible. The series of problems was conducted within various constraints. The results showed that the two groups used different strategies, but the differences were consistent within the groups. The scientists tried out a series of possible combinations to maximize their knowledge of the problem in hopes of discovering a general rule. The architects attempted a design based upon cursory knowledge of the problem; if that was not acceptable, the next most likely solution was tried. In other words, the natural scientists attempted to discover general principles, while architects focused on desired solutions.

In spite of the agreement that design sciences differ from natural science, there has been difficulty in finding a common description of the process of design. Two classical theories are those of Simon (1981) and Alexander (1964). Simon conceived of design as an instance of problem-solving. A formal description of problem-solving involves representing the problem as a problem-space with initial, intermediate, and goal states. The solution to the problem involves searching for operators which will transform the initial state into the goal state. For ill-structured problems, heuristics rather than algorithms are required to achieve ends. A standard heuristic for the solving of difficult problems is means-end analysis. Ends are defined, and means to those ends are specified. If no means are apparent, the problem is decomposed into a hierarchy of sub-problems. This decomposition continues until means are discovered to solve the sub-problems. Thus, problem solving, and therefore design, is simply a matter of finding the best description of the problem. In this case, a theory of design is equivalent to a formal representation of problem-solving heuristics.

Alexander (1964), an architect, presented a theory of design problem-solving which predated Simon but also relied on representing the problem as a space. Alexander differed from Simon in that he advocated "unselfconscious" problem decomposition rather than the "self-conscious" methods of Simon. Unselfconscious problem solving involves representing design specifications as points in a problem-space and discovering highly interconnected clusters of points. These points represent important factors to be considered. As points are connected into a hierarchy of factors, representations of crucial issues at differing levels of generality emerge. Again design is represented as a matter of finding the best description of the problem space.

The problems with the two theories are summarized by Carroll and Rosson (1985). Both theories combine prescription with description. Both fail to illustrate their theories with anything more than idealized examples. Both reduce design to a problem of finding the correct description of the problem-space. Carroll and Rosson examined empirical studies of designers in action and based on these studies they argue that, contrary to Simon and Alexander, the process is:

- Non-hierarchical
- Neither strictly bottom-up nor top-down.
- Radically transformational, involving the development of partial and interim solutions which may ultimately play no role in the final design.

- Intrinsically the discovery of new goals.

Carroll and Rosson's formulation, while accepting Simon's distinction between science and design, does not accept his characterization of the design process and emphasizes its complexity and unpredictability. This complexity and unpredictability presents a dilemma from a pedagogical perspective. If design is too complicated to be represented, how can it be communicated? One solution is to use idealized process models.

There have been many attempts to model the design process, the earliest dating back over 20 years (Asimow, 1962; Jones, 1963). Although these models came out of engineering and architecture, which may seem more predictable than instructional design, these authors emphasize the complexity and uncertainty of design. Indeed, Asimow asserts that philosophy and ethics are part of engineering so there can be no relying on purely empirical principles. In spite of these complexities, Asimow, Jones, and others constructed design models. These models typically represent design as a phased-state development process with or without "feedback loops. All traditional phased- state models of the design process reduce to three stages-analysis, synthesis, and evaluation as first presented by Jones (1963). More complex models, such as **ISD** models, either elaborate the three main stages or add pre- and post-design processes (cf., Andrews and Goodson, 1980). For the most part, these models have not been tested empirically, nor have their originators felt obliged to do so. The justification of such models is primarily pragmatic rather than theoretical. It is to reduce error and delay and to allow more imaginative and advanced designs (Jones, 1963). As Broadbent (1973) has pointed out, however, these models do not really define a design process as much as a decision sequence. They simply assert that it is more useful to make certain decisions before others are potentially testable hypothesis.

In spite of Simon's clarification of the distinction between the natural and artificial sciences and the lack of empirical or theoretical underpinning to most design models, many current models of design claim to emulate a scientific approach, the systems approach. As Kemp asserted, "The systems approach is based on the method of scientific inquiry. ." (quoted in Nunan, 1983, p. 51). This justification suffers from several defects. First, it confuses science and design. Asimow (1962) noted that, " The end of [scientific] research is a finding which will be true in many situations; of design, a piece of hardware" (p. 48). In the field of education, Ausubel (1959) also pointed out the distinction between scientific research and design research and warned of the problems associated with confounding the two.

In addition to the confusion of design and science, the pragmatic justification presupposes a naive theory of scientific activity based upon "the scientific method. Nunan (1983) notes that this account of the scientific method is a textbook version of scientific activity which Kuhn (cited in Nunan, 1983) criticized:

> Inevitably, however, the aim of such books is persuasive and pedagogic; a concept drawn from them is no more likely to fit the enterprise that produced them than an image of a national culture drawn from a tourist brochure or a language text (p. 84).

Even those who argue that scientific reasoning can be formally modeled (e.g., Langley, Simon, Bradshaw, & Zytkow, 1987) admit that there may be no one scientific method. Given the distinctions between science and design, the general state of uncertainty of what science is (Bechtel, 1988), and the empirical evidence of actual design processes, the argument that a systems approach to instructional design is desirable because it is "scientific seems unpromising at best.

Still, most instructional design models attempt to apply general systematic and analytic procedures to instructional situations. But, if instructional design is a process within the realm of the artificial, it would seem more appropriate for designers to focus on solving problems and achieving goals by synthesizing materials in a manner similar to the one the architects used in Lawson's study. As Rowe (1987) pointed out, since problem solvers are rarely in a position to identify all possible solutions, they must deal with bounded rationalities. *Bounded rational*ity refers to the need to make decisions without complete information~. Decision making without adequate information is typical of design. Indeed, Schön (1988) has argued that defining characteristics of design activities are *uncertainty, uniqueness,* and *conflict.* In this light, design becomes a process of reflection- in-action; and designers take on the task of turning indeterminate situations into determinate ones (Schön, 1987).

The nature of design, Lawson (1980) argued, is that problems cannot be comprehensively stated, and that any statement of a problem requires subjective interpretation on the part of the designer. Solutions are uncountably large in number and there is never one that is optimal. The design process is endless, with no infallibly correct methodology. In fact, Alexander (1964) argued that if there were an algorithmic methodology, the process could no longer be called design. Design is a prescriptive activity which involves value judgments on the part of the designer, who works in the context of a need for action. In sum, the limits of analysis are determined by the fact that complex problems are subjective and cannot be exhaustively analyzed. Therefore, design begins by being a conjecture, and after utilization, a modification job which involves finding as well as solving problems (Lawson, 1980). Given this conceptualization of the design process, it follows that any design methodology which acknowledges the complexity of the situation may be more efficient because it anticipates and short-circuits the kinds of problems designers typically encounter.

SIMILARITIES BETWEEN SOFTWARE DESIGN AND INSTRUCTIONAL DESIGN

Engineering and education are both disciplines which fit Simon's definition of artificial sciences. Software design and instructional design are fields that have similar methodologies and purposes. The waterfall model (Maher & Ingram, 1989) of software design and the interservices ISD model (Branson, 1975) represent two well-known models from the respective fields. Both models consist of five steps. The waterfall model includes Analyze, Design, Implement, Test, and Maintain. The interservices ISD model specifies Analyze, Design, Develop, Implement, and Control. The superficial similarities are obvious. At a deeper level, Maher and Ingram (1989) note that in both fields, designers attempt to be systematic in approaching large, complex problems. Designers in both fields attempt to bring orderly and replicable practices to disciplines which are dominated by individual practitioners. Both have typically advocated the use of formative evaluation procedures in the development of systems. Additionally, the two often deal with similar constraints in planning, budgeting, scheduling, and tracking the development of materials.

The most fundamental difference between the two fields is the degree of rigor that can be expected in each. Software designers deal with systems that are based on mathematical logic. Instructional designers deal in part with computer software, but primarily with systems based on human cognition, which entail more uncertainty and accept more ambiguity.

Based on the large number of similarities and the minor differences that exist, practitioners in the two fields have often used similar models in their efforts to create effective materials. Indeed, Maher and Ingram (1989) have asserted that instructional designers could benefit from studying the methods of software designers. One method of software design which has

been widely endorsed recently (Jordan, Keller, Tucker, & Vogel, 1989; Luqi, 1989; Schneiderman, 1987; Tanik & Yeh, 1989; Whitten, Bentley, & Barlow, 1989) is called *rapid prototyping.*

RAPID PROTOTYPING

Recently, capitalizing on the increased capabilities of software development tools, software designers have begun to use the design methodology called *rapid prototyping ping.* Figure 1 shows a model of rapid software prototyping based upon Lantz (no date). Rapid soft ware prototyping has been defined by Lantz (no date) as a ". . . system development methodology based on building and using a model of a system for designing, implementing, testing and installing the system (p. 1). In this methodology, after a succinct statement of needs and objectives, research and development are conducted as parallel processes that create prototypes, which are then tested and which may or may not evolve into a final product. The rapid prototype should include any required database, the major program modules, screen displays, and inputs and outputs for interfacing systems. To perform the prototyping process, it is necessary to have physical and logical definitions of the system, an opportunity to exercise the prototype, and software which allows the rapid building and modification of the prototype. In Lantz's terminology the physical and logical definitions correspond approximately to an instructional strategy and instructional objectives. It should be noted, however, that the definitions are a product of the prototyping process, as can be seen in Figure 1. The initial definitions serve only to construct the model of the system. It is through the rapid prototyping process that initial definitions evolve into final definitions.

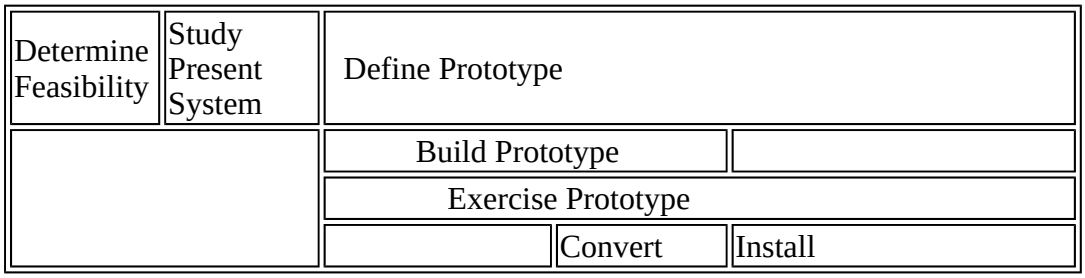

Figure 1. Prototyping Approach to Software Design

Although the term *rapid prototyping* is new, the underlying methodology is not. In hardware engineering, the use of prototypes as a way of testing ideas has a long and successful history. The image of model airplanes in wind tunnels is familiar to all. Dreyfuss (1974) recommended the use of mock-ups and user-testing as essential to the design process. Asimow's (1962) *Introduction to Design* specifically mentions the use of prototypes as an empirical methodology. Wilson and Wilson (1965) also describe prototyping as a design methodology. This tradition has evolved into modern systems analysis techniques such that Whitten, Bentley, and Barlow's (1989) textbook, *Systems Analysis* & *Design Methods,* integrates prototyping into the standard model. The use of rapid prototyping in software engineering is essentially the extension of a successful design methodology into a new domain.

The use of rapid prototyping in software design depends on development software which allows rapid construction and modification of software. As anyone familiar with computers knows, software development has been a tedious and time-consuming procedure. The

extreme time penalties involved in modifying software under traditional conditions obliged software developers to thoroughly specify product characteristics before a project was actually coded. The advent of various powerful and modular software prototyping tools has allowed the prototyping methodology to be applied to a domain where previously it was impractical. Thus the use of rapid prototyping in software design is a function of the development media available.

The motivation to use rapid prototyping is based upon both faults in the traditional development process and advantages founds with prototyping. Some of the faults with traditional methodologies which Lantz (no date) has documented are as follows:

- They are thorough but don't please users.
- They produce extensive documentation but don't reduce communication problems.
- They identify phases but don't decrease project time.
- They describe the system thoroughly but don't guarantee it's the right system.
- They delineate the skills needed but don't cut human resource needs.
- They track project costs but don't reduce them.

Also, Maher and Ingram (1989) assert that the advantages of rapid prototyping are that it allows users to try out the system, discover the problem areas, and have input into the selection of an appropriate interface. Lantz (no date) provides evidence that rapid prototyping pleases users, reduces development costs, decreases communication problems, lowers operations costs, slashes calendar time, and produces the right system for the designated task.

Given the similarities between software engineering and instructional design, especially instructional design for computer- based instruction, rapid prototyping may offer all the same advantages in instructional development that it offers in software development, The argument can be made that rapid prototyping is even more appropriate for instructional design because it allows the flexibility needed when dealing with the greater complexity of a human factors-intensive field such as the process of instruction.

A MODEL OF RAPID PROTOTYPING

The word *model* is widely used and frequently undefined. We follow Marca and McGowan (1988) and define M to be a model of a process P if M answers questions about P with accuracy A. It should be pointed out that accuracy A is not absolute accuracy. Therefore, our model does not represent rapid prototyping completely. We concur with Carroll and Rosson (1985) that, "Design is a process, it is not a state and cannot be adequately represented statically (p. 27).

Figure 2 represents the events that occur in a rapid prototyping environment, when prototyping is specifically used as a method for instructional design. The overlapping boxes are meant to represent the fact that the various processes do not occur in a linear fashion. In other words, the analysis of needs and content depends in part upon the knowledge that is gained by actually building and using a prototype instructional system.

Assess Needs & Analyze Content	Set Objectives
	Construct Prototype (Design)
	Utilize Prototype (Research)
	Install & Maintain System

Figure 2. The Rapid Prototyping Model

As with software development, rapid prototyping in instructional systems design is the building of a model of the system to design and develop the system itself. The process begins, as in most traditional instructional design models, with the analysis of needs and content and a statement of tentative objectives. The statement of objectives at this stage is simply the definition of a plan for instructional design. As a plan, it serves two functions (Streibel, 1989): to communicate to everyone involved the purpose of instruction and to delineate tasks the learner ~will pursue. Rapid prototyping continues with the parallel processes of design and research, or construction and utilization. It is assumed that full understanding of needs, content, and objectives is a result of the design process and not an input into it. Reigeluth (1989) notes the expediency of coupling design with research. He suggests that separating research from design, as has often been done in the past, is not the best manner by which to build prescriptive theory. If a designer who is familiar with theory incorporates it when designing products and studies its application when conducting product evaluation, research and development can be very effective as parallel processes. Minimally, research should be conducted to discover the complexities of the subject matter, prerequisite knowledge needed to understand the content, and the presentation modes that are most conducive to acquiring the material.

DIFFERENCES BETWEEN RAPID PROTOTYPING AND TRADITIONAL ISD

Although there are traditional models which resemble rapid prototyping, both in their sequencing of decisions and their use of prototypes (e.g., Sullivan, 1971), the orientation of the rapid prototyping approach is to acknowledge rather than to minimize the com plexity of actual situations. The emphasis of many writers (e.g., Briggs, 1977) is on instructional planning, rather than learning from actual situations. Needs assessment and serious field testing are minor topics. No doubt, this is partly a function of typical instructional design situations, which may constrain these activities, but it also represents an attitude of technical rationality which discounts the interacting complexity of budget, time, content, methods, local history, talent, and social interaction. As a result, many traditional models emphasize early constraining of design decisions, while rapid prototyping follows the pragmatic design principle of minimum commitment (Asimow, 1962; Wilson & Wilson, 1965), that at each stage in synthesizing a design no commitment is made beyond what is absolutely necessary to solve the problem at hand. In fact, given the power of our available tools, it is not outside the spirit of rapid prototyping to create alternate, and even contradictory designs, as has been advocated by Carroll and Rosson (1985). Traditional instructional designers may find this suggestion surprising, but the fact that contradictory .approaches may have efficacy is illustrated by the work of Asher and Gattegno in language teaching. Asher (1977) advocates a methodology in which *only* the teacher speaks. Gattegno (1972), however, originated a methodology in which the teacher *almost never* speaks. Both methods have generated sufficient success to produce a following. It is not unthinkable that the generation and testing of seemingly contradictory designs may result in theoretical knowledge. It is well-established (Ellul, 1964) that the natural sciences have often been advanced by work in the artificial

sciences.

USING THE DESIGN

A crucial part of the prototyping process is the utilization of the design with potential learners. Utilization is the situated action in which the learner develops cognitive skills and learns content. During utilization, the designer observes the learner and asks questions to discover strengths and weaknesses of the prototype. As a result of the utilization phase, the learner and the designer have separate learning experiences which are determined by their individual plans and their reflections about and cognitive reconstructions of the utilization experience. Both the learner and the designer are affected by the utilization phase in that they gain new information by problem solving, but additionally and more important, utilization involves problem discovery. For the designer, the discovery of new problems results in the modification of the tentative objectives or the creation of new ones. With these objectives, the rapid prototyping process begins again. For evaluation, we believe that the detailed observation and debriefing of a small number of subjects can be revealing (Komoski, 1974). The end of a design project is an appropriate artifact not a generalization. An instructional system must be adapted to a unique situation and need not have general applicability.

USING RAPID PROTOTYPING IN INSTRUCTIONAL DESIGN

Rapid prototyping presupposes a design environment which makes it practical to synthesize and modify instructional artifacts quickly. Without such an environment it becomes inefficient and, therefore, loses its attractiveness. To make prototyping efficient and effective, certain types of media are required. Rapid prototyping requires the availability of tools (mainly computer software) that offer *modularity* and *plasticity*. Modularity allows a segment of the instructional unit to be added, removed, or modified without affecting severe interactions in the other segments or the unit as a whole. Examples of modular media are loose leaf notebooks, overhead transparency presentations, and object-oriented computer programs such as HyperCard. The second requirement, plasticity, refers to the ability to change aspects of a unit of instruction with only minor time or cost penalties. Plasticity is difficult to achieve with most types of instructional media. Textbooks, film, videodiscs, slides, audio recordings, and even transparencies are all created using technologies which make revision tedious or costly after the product is initially mastered. Again, computer programs such as HyperCard offer a high degree of plasticity for instructional design. Thus, it is probable that only in the context of computer-based instruction is rapid prototyping a viable methodology. Although rapid instructional prototyping can always be accomplished given a sufficient commitment of money and human resources, it has become a practical instructional design methodology only within the modern software development environment.

INSTANCES FOR USE OF THE RAPID PROTOTYPING MODEL

While we do contend that the rapid prototyping model, when feasible because of the availability of modular and plastic media, is more compatible with real-world design processes than are traditional models of instructional design, we do not mean to suggest that the existing body of knowledge in the field of instructional design be disregarded. Indeed,

the experience of generations of instructional designers and researchers constitutes a body of situated cognition that serves as a platform for further design activities. In addition, traditional analytic approaches offer a good first step in the design process. Neither do we assert that use of the rapid prototyping model would be appropriate in every instructional situation. Many situations, such as the production of satellite-broadcast lecture courses, make rapid prototyping a near impossibility. It appears, however, that in certain circumstances, especially when coupled with modular software development tools, rapid prototyping is a plausible model for instructional design. We believe that rapid prototyping appears to be appropriate in at least the three following types of situations: cases that involve complex factors which make prediction problematical, cases where we have experience but lack satisfaction with results derived from conventional methods, and new situations where there is not an abundance of experience from which to draw.

Cases with complex factors

There are two reasons why rapid prototyping appears to be an appropriate methodology in cases where complex factors make prediction problematical. The first reason focuses on the nature of complex factors. In learning situations, complex factors typically concern either communication problems such as human-machine interaction, cognitive processing capabilities such as higher order thinking skills, or "soft skills such as management skills where there is really no well-defined body of knowledge to guide us. In dealing with these factors, a model of instructional design is required that can provide plasiticity and modularity to allow for the variations that occur in each new situation of use. Flexibility of the instructional system is also the key to dealing with situations in which prediction is problematical. In such situations, rapid prototyping is more appropriate than traditional models of instructional design because it is not based on general principles that standardize every learning situation by forcing them all into similar molds. Problematical prediction is less of an issue in the rapid prototyping model because front-end analysis is only intended to be a beginning point. Plans can easily be changed during the research, development, and even utilization phases because the model takes advantage of the flexibility of the medium used to create the instructional sequence and strategy.

When Conventional Methods Yield Unsatisfactory Results

Streibel (1989) wrote about the challenge that Suchman's theory of situated learning presents for instructional designers. In that pa; per, Streibel expressed his own feelings of frustration with the inadequacy of traditional instructional models. He wrote:

> I first encountered the problematic relationship between plans and situated actions when, after years of trying to follow Gagné's theory of instructional design, I repeatedly found myself, as an instructional designer, making ad hoc decisions throughout the design and development process. At first, I attributed this discrepancy to my own inexperience as an instructional designer. Later, when I became more experienced, I attributed it to the incompleteness of instructional design theories. Theories were, after all, only robust and mature at the end of a long development process, and instructional design theories had a very short history. Lately, however, I have begun to believe that the discrepancy between instructional design theories and instructional design practice will never be resolved because instructional design practice will always be a form of situated activity (i.e., depend on the specific, concrete, unique circumstances of the project I am working on). Furthermore, I now believe instructional design theories will never specify my design practice at anything other than the most

general level. (Streibel, 1989, p. 7)

Certainly, there are many cases in which traditional models have worked satisfactorily in achieving prescribed instructional objectives. There are also many instances, however, in which traditional models have not done the job expected of them, and it is in these cases that rapid prototyping might be seen as a viable alternative to conventional practice.

There are a number of reasons why traditional models may not be successful. As Streibel suggested, they may be incomplete, or they may not account for the situated nature of knowledge. Or perhaps, as Maher and Ingram (1989) suggested, many traditional designs have a linear quality which in many instances is not a true reflection of the design process. They argue specifically that recent research has found instructional design models with sequential, hierarchical features do not adequately represent what people really do, or what they should do, in specific design situations, and that software engineers frequently need more realistic, more flexible models to follow in planning and executing a project. Conklin and Bridge land (1986) support this position and assert that, "It has become a commonplace that people don't *really* proceed along the linear stages of the waterfall model. The same could be said about instructional design models, with the possible exception of large projects where different people are responsible for each stage of the project.

Reigeluth (1989) recognized that educational technology is a field that is rapidly changing, and he discussed the challenges that face theorists and practitioners because of changes that are occurring in the field. He suggested that some of the most important new directions will include, among other things, the development of prescriptions for types of learning which have been largely ignored by the field (such as situated learning), and development of prescriptions that take advantage of the unique capabilities of new technologies. Although he was referring to instructional strategies, rather than design strategies, we contend that the argument holds for both.

Unfamiliar Situations

Rapid prototyping appears to be more appropriate for use than traditional methods of instructional design in situations, such as learning from hypertext, where there is little experience from which to draw. This is because in the rapid prototyping methodology, research is conducted concurrently with development; therefore little formal research is needed to begin a project, and much information can be gathered from research conducted as learners use the prototype. The rationale for rapid prototyping recognizes that, in reality, each learning situation is to some degree different from any before or after, and therefore acknowledges that all research is in some manner relative to the situation in which it was conducted. Thus, rapid prototyping is designed so that each learning situation is dealt with as a new situation, with unique problems to be discovered and solved.

RAPID PROTOTYPING AS A PARADIGM SHIFT IN INSTRUCTIONAL DESIGN

Up to this point, the purpose of this paper has been primarily to point out the features of the design environment that make rapid prototyping a plausible model of instructional design. This has included an explanation of how the difference between design sciences and natural sciences influences the kind of design models we should use, and a statement of the situational nature of knowledge, which should affect the type of design procedures we

employ. Additionally, we have attempted to give a brief introduction to the terms, stages, and operations used in rapid prototyping, so that readers may have some understanding of the practical aspects of this methodology.

At this point, we will attempt to delineate assumptions which we believe make rapid prototyping more than just an alternative model of instructional design. Based on these assumptions, we believe that the rapid prototyping methodology represents a paradigmatic shift in understanding the nature and purpose of the field of instructional design.

First, based on the arguments given in the first section of this paper, we have assumed that there is a legitimate and important difference between science and design. Second, we have noted our belief that it is possible to acquire specific knowledge of the world by using materials synthesized within the realm of design science. Third, we assume that there is a fundamental difference between the meaning of validity when applied to design theories, as opposed to the manner in which it is used in educational psychology. The difference, according to Reigeluth (1989), is based on the purposes of the study. When attempting to add to a knowledge base that is descriptive or analytical, construct validity should be the major concern of research. But as in the case of instructional design, where research is aimed at prescription and synthesis, "optimality becomes the major research concern. The focus on optimality might be best described as research conducted to determine if the theory or the model used achieves the desired results in a specific instructional situation. As Reigeluth outlines, the process for determining optimality requires (a) using one particular model to produce an instructional product, (b) conducting a series of formative evaluations in naturalistic conditions using both obtrusive measures such as face-to-face interviews and unobtrusive measures such as observation, and (c) replicating the study using different content, learners, settings, and mediation as dependent variables. Reigeluth suggests that "this kind of study yields much more data about a broad range of features of the theory or model, and these data are far more relevant for improving the theory or model than any experimental study (1989, p. 72). Research of this type has recently been reported by Ingram (1988). Although these researchers were primarily concerned with design prescriptions rather than designs, the methodology of rapid prototyping is amenable to both endeavors. This is because design prescriptions may be thought of as designs themselves (Perkins, 1986) and therefore may be rapidly prototyped.

Finally, we assume the post-positivist position that human experience is a subjective entity and that perfect objectivity is not achievable, at least in regard to human affairs. In this paper we refer specifically to design, which is a form of knowledge construction. Documentation for this particular assumption could lead to volumes in itself; we acknowledge that we have neither the space nor the resources to present the complete argument here. For those who seek a very complete and detailed argument for the case of post-positivism, especially in the behavioral sciences, we suggest Guba and Lincoln's *Fourth Generation Evaluation,* (1989). If this position is valid, it means most importantly that there is no one "right way for learners or designers to acquire knowledge since there is not one particular set of knowledge claims that can be accepted as truth. Since the process of design is a process of knowledge acquisition, the implications of this position are several. Asimow (1962), writing about engineering design, emphasized the conflicting demands of uniqueness and uncertainty, and that, since philosophy is based upon what we believe, there cannot be one philosophy of design. If engineering design is uncertain and subjective, instructional design must be even more so. Recent theoretical writing on the design process (Carroll & Rosson, 1985; Schön, 1988) has not supported the technical rationality approach to design advanced by Simon and has emphasized the subjective nature of skilled design. Such theories of design undermine traditional instructional design models which strive for technical rationality, unless it be

asserted that traditional design is not a member of the general category, design-an awkward position.

When based on the four assumptions cited above, use of the rapid prototyping methodology becomes more than just the acceptance of a viable alternative to instructional design, it becomes a statement of belief about how design takes place and how instructional designers can synthesize learning environments. Not every environment will be ame nable to this methodology. The final test of rapid prototyping, like anything in the design sciences, is not whether it is based on true assumptions, but whether it is useful. It has proved to be useful in other domains.

AN EXAMPLE

One of our students had been involved in the development of a computer-based grammar tutor for foreign students. He found that his analysis of the types of information which would be useful to students, the types of feedback that should be given, and the general structure of the tutorial was becoming very complex. On the other hand, he felt that he could not pilot-test the program until it was in a relatively finished form. I suggested to him that he should try rapid prototyping the tutorial. The prototype was deliberately only a model of the finished product. That is, it contained only the major elements of the final tutorial, and these elements were presented in a schematic way. He was able to produce the prototype in a short period of time (a matter of hours) and immediately started testing it with potential users, while collecting their suggestions and comments. He reported that this process answered many of his questions and that he was able move quickly toward a full version of the tutorial. This example illustrates the essential features of rapid prototyping. First, a model of the system was used to investigate and design the full system. Second, the software environment allowed rapid synthesis and modification of the system. Third, a slow and uncertain process of analysis and detailed specification was replaced by an efficient process of hands-on design. Although this application was successful, it depended upon two factors: a plastic and modular medium, and an intention to learn through the process of design.

PROS AND CONS

Some may argue that rapid prototyping is nothing new-the methodology of rapid prototyping has always been with us, even if the models of design did not acknowledge it. In one sense this is true. Where possible, engineering design and instructional design have used prototypes. The use of prototypes is not the same as rapid prototyping, however. In many cases, the use of prototypes is dictated by the severe consequences of error (i.e., aircraft design), rather than efficiency considerations. Rapid prototyping emphasizes the rapid synthesis and utilization of designs because the medium affords it.

Others may say that traditional models with their formative evaluation are a kind of prototyping. First, the need for formative evaluation is an acknowledgment that front- end analysis, no matter how rigorous, cannot guarantee a successful design. Second, prototyping is not rapid prototyping. Traditional models do not emphasize the efficiency potential of modern software environments. The prototypes produced in such a methodology are really pilot tests. They represent a relatively final form of the instructional system. Rapid prototyping differs in its assumptions from formative evaluation approaches. Perhaps most importantly, rapid prototyping assumes that design involves the discovery of goals as well as

their satisfaction. To discover goals, rapid prototyping places synthesis before analysis, or uses an analysis-by- synthesis approach. The reasoning behind this approach stresses that uncertainty, uniqueness, and conflict are the defining attributes of design situations; design, therefore, is treated not only as problem-solving, but as *making* (Schön, 1988). Seeing design as making means that this approach shares many fundamental ideas with Winograd and Flores (1987) and Suchman (1987). These ideas emphasize that design is highly contextualized and not a product of technical rationality. Instructional designs are negotiated products, based upon many factors besides learning theory and instructional prescriptions. Differences like this make rapid prototyping a legitimately alternative approach with its own set of advantages and disadvantages.

Whitten et al., (1989) have summarized advantages and disadvantages of rapid proto typing in a systems engineering context. We have adapted their conclusions into the instructional design environment. The following are potential advantages of rapid prototyping:

- It encourages and requires active student participation in the design process.
- Iteration and change are natural consequences of instructional systems development. Clients tend to change their minds.
- Clients don't know their requirements until they see them implemented.
- An approved prototype is the equivalent of a paper specification-with one exception. Errors can be detected earlier.
- Prototyping can *increase* creativity through quicker user feedback. (But see below.)
- Prototyping accelerates the development cycle.

The main disadvantage of prototyping can be summed up in one complaint that is easy to imagine: it has a tendency to encourage informal design methods which may introduce more problems than they eliminate. This failure can be avoided if the following issues are kept in mind:

- Prototyping can lead to a design-by-repair philosophy, which is only an excuse for lack of discipline.
- Prototyping does not eliminate the need for front-end analysis. It cannot help if the situation is not amenable to instructional design.
- A prototype cannot substitute completely for a paper analysis.
- There may be many instructional design problems which are not addressed by prototyping.
- Prototyping may lead to premature commitment to a design if it is not remembered that a design is only a hypothesis.
- When prototyping an instructional package, creeping featurism (the adding of bells and whistles) may lead to designs that get out of control.

- Prototyping can *reduce* creativity by eliminating the urge to find better designs.
- Prototyping environments can lead to designs that execute less efficiently than designs instantiated in dedicated authoring languages.

SUMMARY

We have presented several arguments for the notion that rapid prototyping is a viable model for instructional systems design in a computer-based instruction context. First, there is a long history of the successful use of prototyping in software engineering. Additionally, there are strong similarities between software engineering and instructional systems design. Rapid prototyping is compatible with the evidence of empirical research on how designers work. What's more, rapid prototyping is not based on naive models of the scientific method. It is only recently that software authoring tools have made rapid prototyping realizable. Rapid prototyping is consistent with current views of how instructional design research should proceed. Finally, it is a methodology which can survive more sophisticated theories of design. That is, it allows us to be systematic in the face of design theories which emphasize the representational complexity and situational grounding of design. Previous models were based upon naive idealizations of how design takes place. When naive theories are replaced with more sophisticated ones the intellectual underpinnings of **ISD** are weakened. We then must choose between teaching unjustified models or no models at all. Rapid prototyping bridges this dilemma.

REFERENCES

Alexander, C. (1964). *Notes on the synthesis of form.* Cambridge, MA: Harvard University.

Andrews, D. H., & Goodson, L. A. (1980). A comparative analysis of models of instructional design. *Journal of Instructional Development, 3(4),* 2- 16.

Asher, J. J. (1977). *Learning another language through actions-The complete teacher's guidebook.* San Jose, CA: Sky Oaks Productions.

Asimow, M. (1962). *Introduction to design.* Englewood Cliffs, NJ: Prentice-Hall.

Ausubel, D. P. (1959). Viewpoints from related disciplines: Human growth and development. *Teachers College Review, 60,* 245-254.

Bechtel, W (1988). *Philosophy of science.* Hillsdale, NJ: Lawrence Erlbaum.

Branson, R. K. (1975). *Interservice procedures for instructional systems development: Executive summary and model.* Tallahassee: Florida State University, Center for Educational Technology. (ED 122 022)

Branson, R. K., & Grow, C. (1987). Instructional systems development. In R. Gagne (Ed.), *Instructional technology: Foundations* (pp. 397-428). Hillsdale, NJ: Lawrence Erlbaum Associates.

Briggs, L. J. (Ed.). (1977). *Instructional design.* Englewood Cliffs, NJ: Educational Technology Publications.

Briggs, L. J., & Wager, W W (1981). *Handbook of procedures for the design of instruction.* Englewood Cliffs, NJ: Educational Technology Publications.

Broadbent, C. (1973). *Design in architecture.* London: John Wiley & Sons.

Carroll, J. M., & Rosson, M. B. (1985). Usability specifications as a tool in iterative development. In H. R. Hartson (Ed.), *Advances in human-computer interaction* (pp. 1-28). Norwood, NJ: Ablex.

Carroll, J. M., & Rosson, M. B. (1987). Paradox of the active user. In J. M. Carroll (Ed.), *Interfacing thought* (pp. 80-111). Cambridge: MIT Press.

Conklin, J., & Bridgeland, D. (1986). *Beyond macro- iteration: An organic model of system design. (STP398-86).* Austin, TX: MCC Software Technology Program.

Dreyfuss, H. (1974). *Designing for people.* New York: Grossman Publishers.

Ellul, J. (1964). *The technological society* (I. Wilkinson, trans.). New York: Knopf.

Gagne, R. M. (Ed.). (1987). *Instructional technology: Foundations.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Gagne, R. M., & Briggs, L. J. (1979). *Principles of instructional design* (2nd ed.). New York: Holt, Rinehart and Winston.

Gattegno, C. (1972). *Teaching foreign languages in school: The Silent Way (2nd ed).* New York: Educational Solutions.

Cuba, E. C., & Lincoln, Y. 5. (1989). *Fourth generation evaluation.* Beverly Hills, CA: Sage.

Ingram, A. L. (1988). Instructional design for heuristic-based problem solving. *Educational Communication and Technology Journal, 36,* 211-230.

Jones, J. C. (1963). A method of systematic design. In J. C. Jones, and D. C. Thornley (Eds.), Con*ference on design methods* (pp. 53-73). Oxford: Pergamon.

Jordan, P. W, Keller, K. S., Tucker, R. W, & Vogel, D. (1989). Software storming. *Computer,* 22(5), 39-48.

Komoski, P. K. (1974). An imbalance of product quality and instructional quality: The imperative of empiricism. *AV Communication Review, 22,* 357- 386.

Langley, P, Simon, H. A., Bradshaw, G. L., & Zytkow, J. M. (1987). *Scientific discovery.* Cambridge, MA: MIT Press.

Lantz, K. E. (no date). *The prototyping methodology.* Englewood Cliffs, NJ: Prentice Hall.

Lawson, B. (1980). *How designers think.* Westfield, NJ: Eastview Editions.

Luqi. (1989). Software evolution through rapid prototyping. *Computer 22(5),* 13-25.

Maher, J. H., & Ingram, A. L. (1989, February). *Software engineering and ISD: Similarities, complementarities, and lessons to share.* Paper presented at the annual meeting of Association for Educational Communications and Technology, Dallas, TX.

Marca, D. A., & McGowan, C. L. (1988). *SADT™ Structured analysis and design technique.* New York: McGraw-Hill.

Nunan, T. (1983). *Countering educational design.* New York: Nichols Publishing.

Perkins, D. N. (1986). *Knowledge as design.* Hills dale, NJ: Erlbaum.

Reigeluth, C. M. (1989). Educational technology at the crossroads: New mindsets and new directions. *Educational Technology Research and Development, 37,* 67-80.

Romsizowski, A. J. (1981). *Designing the user interface.* Reading, MA: Addison-Wesley.

Schön, D. A. (1987). *Educating the reflective practitioner: Toward a new design for teaching and learning in the professions.* San Francisco: Jossey-Bass.

Schön, 0. A. (1988). Designing: Rules, types and worlds. *Design Studies, 9,* 181-190.

Simon, H. A. (1981). *The sciences of the artificial* (2nd ed.). Cambridge, MA: MIT Press.

Streibel, M. J. (1989, February). *Instructional plans and situated learning: The challenge of Suchman's theory of situated action for instructional designers and instructional systems.* Paper presented at the annual meeting of Association of Educational Communications and Technology, Dallas, TX.

Suchman, L. A. (1987). *Plans and situated actions: The problem of human-machine communication.* New York: Cambridge University Press.

Sullivan, H. J. (1971, July). Developing effective objectives based instruction. *Educational technology,* 55-57.

Tanik, M. M., & Yeh, R. T. (1989). Rapid prototyping in software development. *Computer* 22(5), 9-10.

Whitten, J. L., Bentley, L. D., & Barlow, V. M. (1989). *Systems analysis & design methods* (2nd ed.). Homewood, IL: Irwin.

Wilson, I. G., & Wilson, M. E. (1965). *Information, computers and system design.* New York: John Wiley & Sons.

Winograd, T., & Flores, F. (1987). *Understanding computers and cognition.* Reading, MA: Addison-Wesley.