3. Design and cognition as inquiry

3.1 The nature of design

In this chapter I will begin to present my alternative theory of design activity. As will become evident, it draws on the analysis by Schön (e.g. 1983, 1987), which in turn owes a great deal to Dewey's theory of inquiry (1938, 1949), both of which will be discussed later in this chapter.

The view of design as problem solving has been criticized many times (e.g. Holt 1985), often on the grounds that design does not fulfill the conditions of problem solving theory (e.g. Rittel 1973). Take the earliest stages of the design process as an example. On the one hand, there is the conventional view of this as problem solving by pure analysis, based on the view of the design problem as being given. On the other hand, there are the conditions of actual designing, where design problems are anything but given, so that the designer must also do "problem setting" (Schön 1983 coined this term as a contrast to problem solving), which more or less amounts to producing also the problem itself.

The need for problem setting leads to the impossibility of separating problem definition from problem solving, which in turn provides the basis for the argument that in genuine cases, design cannot be separated into stages. The full argument constitutes the *theory of inquiry*. At the end of the chapter I contrast the resulting characterization with the stage models from chapter 1, showing how it can explain why these fail under genuine circumstances.

The design problem as given

In the conventional view, the design problem is considered as given. Most oftenly this is not stated explicitly, but is left as an unspoken assumption, as if presupposing that the problem exists before design begins (as a specification of what the problem consists in). Note how e.g. "starting from a global statement of a problem" implies that the problem is given when design begins (from Jeffries *et al.* 1981):

> The task of design involves a complex set of processes. *Starting from a global statement of a problem*, a designer must develop a pre-

cise plan for a solution that will be realized in some concrete way (e.g., as a building or as a computer program). ...

Software design is the process of *translating a set of task requirements* (functional specifications) into a structured description of a computer program that will perform the task. ...

One can think of *the original goal-oriented specifications* as defining the properties that the solution must have. The design identifies the modules that can satisfy these properties. How these modules are to be implemented is a programming task, which follows the design task.

Newell & Simon (1972) however clearly state the problem as given:

To have a problem implies (at least) that certain information is *given* to the problem solver: information about what is desired, under what conditions, by means of what tools and operations, starting with what initial information, and with access to what resources. (p. 73, my italics)

They do so, however, because the enumerated elements are exactly those that must be encoded into their model in advance, in order to make it work at all. As before, design methodologists are closer to reality (Parnas & Clements 1986, p. 253, my italics):

Who writes the requirements document? *Ideally*, the requirements document would be written by the users or their representatives. *In fact, users are rarely equipped to write such a document*. Instead the software developers must produce a draft document and get it reviewed and, eventually, approved by the user representatives.

... Determining the detailed requirements may well be *the most difficult part of the software design process* because there are usually no well-organized sources of information.

Hence, *ideally* problem solving theory would be correct, but in reality, producing the problem is work that the designer must do. And it is not a minor issue; it is on the contrary the *most difficult part* of the work. Note that this part is completely absent from the conventional models, since they consider the problem as given. This is obviously a large oversight that cannot be resolved by making minor adjustments to these models.

Parnas & Clements also list a number of reasons why requirements analysis cannot be isolated into an early, separate stage of the design process, the following two are among them:

1. In most cases the people who commission the building of a software system do not know exactly what they want and are unable to tell us all that they know.
2. ... Many of the details only become known to us as we progress in the implementation. (p. 251)

Here they hint at some of the main points in this chapter. I will however begin to analyze the two contrasting views of constraints.

3.2 Constraints are practical

In the traditional view, constraints are parts of the problem definition, each imposing one restriction on what counts as an acceptable solution, or one *requirement* in terms of requirements specifications, which conversely can be seen as consisting of a list of constraints on the solution. It follows that constraints are regarded as given to the designer, as part of the requirements specification, before design begins. Moreover, that they make the designer's task harder by placing restrictions on her available options.

In reality however, not all constraints originate strictly in the requirements specification. Such is the case for legally imposed restrictions, such as building regulations. These still fit the traditional view in that they are laid down long before the architect begins her work, and definitely in that they make her job more difficult:

Design legislation today may cover anything from the safety of electrical goods to the honesty of advertising or the energy consumption of buildings. ... The architect today must satisfy the fire officer, the building inspector and the town planner and in addition, depending on the nature of the particular project, the housing corporation, health inspectors, Home Office inspectors, the water authority, electricity authority, the Post Office, factory inspectors and so the list goes on. (Lawson 1980, pp. 67–68)

Constraints can be both helpful and flexible

Within the design literature there are abundant examples of ways in which constraints are not the fixed restrictions given in advance that standard accounts portray them as. For example, the customary in-

completeness of requirements specifications means that constraints typically *aren't* given, as mandated by design methodology. Guindon (1990b) gives a number of examples out of her protocols where the designer adds requirements that are not given in the instructions. She argues that the incomplete nature of specifications makes requirements elaboration an important task, because the constraints that the designer adds herself are often essential to a good solution:

> By simulating a Lift scenario, the designer realizes that a user may press a floor button to go in one direction, but once inside the lift, may press a lift button to go in another direction. This test case was not mentioned in the problem statement, yet it is critical for the design of a good control algorithm. (p. 288)

In the traditional view, where constraints impose restrictions, it is hard to see how *adding* constraints can be so helpful. But because specifications encountered in practice typically are incomplete, adding constraints is crucial to yielding requirements that capture the desired functionality: "The ill-structuredness of problems in the early stages of design will require structuring—inferences of new goals and evaluation criteria." (*ibid.*, p. 297)

An even greater anomaly is that designers frequently impose constraints that are neither necessary nor objectively valid. They often apply them for practical (but still very good) reasons, and not from a strict necessity that is inherent in the problem. In one case, one of Guindon's subjects says:

> You would rather not have a single point of failure because if it goes down all the elevators go down. So, I'll start off thinking about a distributed control system... (p. 289)

Guindon comments,

> The designer recognizes from past knowledge with similar systems that 'no single point of failure' would be a highly desirable requirement. However, other designers might have considered low cost or high speed to be more desirable than no single point of failure. (p. 289)

In this case, the designer's professional experience suggests a distributed control solution, and as becomes evident later on, he also already knows how to implement it. Being able to apply a technique he is familiar with is probably a major reason for him to impose this

particular constraint. This helps him to draw upon personal knowledge to structure his design problem, rather than as a constraint that will only make his task more complicated.

Still, having a distributed control system could just as well have been a requirement from a client. This example also shows that constraints often may be seen as belonging to the solution, as much as being part of the problem; once again, being helpful rather than problematic, quite contrary to the traditional view. Accordingly, constraints do not necessarily make the designer's job harder. Guindon sees them as being mainly very helpful (1990b, p. 290):

> ...inferred and added requirements mainly serve two purposes: (1) they lessen the incompleteness and ambiguity inherent in the specification of the requirements; and (2) they decrease the range of possible design solutions by acting as simplifying assumptions. In particular, these inferences contribute to problem structuring. Moreover they effectively guide the search of a solution by pruning a large set of possibilities.

The standard view also claims that constraints "decrease the range of possible design solutions", but that this instead *restrains* the designer instead of helping her.

The source of control principle

The standard view of constraints can be summarized thus: Constraints are restrictions on an acceptable solution that are specified in the instructions given to the designer. They are non-optional (but indeed required) and thus beyond the designer's control. It thereby seems that the notion of constraints is full of contradictions: Is a constraint helpful or a hindrance, is it fixed or optional, is it provided in advance or added during design, and is it given *to* the designer, in the problem definition, or imposed *by* the designer, entirely at her own discretion?

First of all, note that these questions make little sense within the frame of reference of problem solving theory. To such a formal model, the origin or history of its elements (such as a constraint) is of no value in reaching a solution, neither can the theory capture such aspects. And an element cannot be good or bad, only its abstract form matters. In constructing a logical proof or solving a laboratory problem, it provides no help to know who established the facts or rules, or when this was done.

The first step in resolving the apparent contradictions is to acknowledge that in real cases, unlike in theoretical analyses, all of these questions are relevant, and we therefore need to adopt a perspective that will allow us to account for them. The origin of a constraint is a point in case. In practice it does matter, because it determines how rigid or flexible the constraint is. This I would like to call the *source of control* principle: the further away from the designer the source of a constraint is located, the less control of it does the designer have, and the less flexible is the constraint (cf. Lawson 1980).

Legislated constraints are completely rigid

One end of this flexibility scale is represented by the legislator:

> ... no designer would want deliberately to construct a dangerous building. However, often regulations have to be applied in situations which were not predicted when they were framed ... they must be satisfied without question, and cannot be weighed against other factors and considerations. (*ibid.*, pp. 67–68)

This kind of constraint imposed by laws and regulations (concerning fire safety, electricity, plumbing, etc.) are absolute and beyond the designer's influence. The legislation has usually been laid down long before the design process begins, and in an institutional context very remote from the designer's office. Therefore, the designer cannot negotiate a problematic legal constraint if it causes problems. Such a situation is not uncommon, since rule systems such as building regulations must be very generally held and cover a wide range of cases (*ibid.*). Neither are they coordinated with other regulations. It would be impossible to anticipate all possible future situations that may arise, or all the ways in which a norm may come to interact with other regulations. For this reason, the problem arises from such a constraint having been laid down in a context completely detached from the setting in which it will be applied. For the designer, there is no other option than to comply with such a norm, the source of which she has no access to. If a conflict arises, other requirements will have to be compromised instead. They are then compromised in a negotiation of conflicting constraints, not because they are less important, but because they can at all be negotiated.

Client-imposed constraints are somewhat flexible

Requirements that are imposed by clients and users represent a middle ground, being somewhat flexible. A client is guaranteed to have a number of wishes and demands on the product she is paying for. Still, if the designer finds that a requirement or a combination of them necessarily leads to a bad solution, she has the option of negotiating this requirement with the client. In particular, a user-imposed demand will have to be compromised if it conflicts with a more rigid constraint such as a legal norm.

If the client agrees to drop a certain harmful prerequisite, this will probably even lead to a more desirable outcome for herself: If she sees the conflict, she can reassess her needs and find a way out that will lead to a better result than would a blind compliance with the given requirements. After all, much of the purpose of the design process is to come to a better understanding of initially vague and unclear requirements. This understanding should be brought back to the client so that she can use it to reappraise her own needs.

Thus, in all, client-controlled constraints are negotiable and rather flexible, and this circumstance should be used to best advantage. In striving for a good end, negotiating constraints is among the best of the available means.

Designer-imposed constraints are completely flexible

On the flexible end of the scale are the constraints that the designer is completely in control of because she formulates them herself. This is the kind of constraint discussed in Guindon's example above. When her designer made a distributed control system into a major priority, this constraint was not chosen out of necessity or because the client or a law dictated it. Constraints of this kind are completely adaptable; the designer can take a totally pragmatic attitude toward them. She can select them out of her own preferences, and introduce or scrap them as she revalues their usefulness.

Hence, a constraint that is rigid and beyond the designer's control can be very problematic and can truly restrict her range of action. But it may also free her from a range of design decisions, and in that case its rigidity is not a problem. That is also the reason why constrains become so powerful under the designer's own command. A well chosen constraint can be very helpful even though not strictly necessary. By reducing too wide a range of options, it can create structure where the requirements specification is lacking. Because of their reductive function, constraints are often seen as simplifying

assumptions (e.g. Guindon 1990b), and under the designer's control this is true because they can do no harm.

The source of control principle resolves the contradictions

The source of control principle can be rephrased as follows: the less the source of control over a constraint is involved in the design process, the more rigid is the constraint. This is just a clarification of Lawson's principle, where "further away from the designer" is replaced with "less involved in the design process". Although constraints are useful, there is no advantage *per se* in their being absolutely rigid and non-negotiable. A flexible constraint gives the designer the option to renegotiate it if complying with it does not lead to a good result. Such renegotiation is what it means to involve the source of control in the design process.

This is the solution to the apparent contradictions above. The source of the contradictions lay in seeing constraints as fixed and as parts of the requirements specification (or in problem solving terms, as defined by the problem statement). For example, a constraint may work as if it were given and beyond the designer's influence—but that is just the special case where the source of control precludes a constraint from being negotiated. So, these are the answers to the above questions: *Is a constraint fixed or optional?* That depends on how much the source of control is involved in the design process. *Is it provided in advance or added during design?* That also depends on the source of control. *Is it given in the problem or imposed by the designer?* It is never strictly "given", as in the traditional view. Different sources of control can make it anything from completely rigid *as if* it were given (but only as if), to designer-elected and thus entirely flexible. *Is it helpful or a hindrance?* In general a constraint is useful, by reducing complexity and adding structure. But if the effect is misdirected and the control of a bad constraint is also beyond the limits of negotiation, then it may become a hindrance.

3.3 Pragmatism & the theory of inquiry

The two perspectives on constraints that I have contrasted belong to two different modes of scientific explanation. Theories that adhere to the traditional perspective typically have nothing to say about the origins of constraints, but this in effect implies that this origin has to be unproblematic—that they have to be "given". This is the case for example when it is said that design *begins* with requirements analysis—implying that no work is required before that—then they must in effect have an independent existence, prior to design or any other activity producing the requirements, etc. And this is even more clear in problem solving theory, where everything is encoded in the problem space before the whole process starts, it is all but stated that the constraints thereby are given, among other things, to the solver.

The philosophical consequences that lie implicit in the standard view of constraints make it a so-called *realist* position. This term refers to their being "real", i.e. existing independently of someone having to "create" them; Dewey (e.g. 1949, Appendix III) used the term "ontological" instead of real, since the very idea of ontology rests on a separation between the knower and the known (the object of knowledge). I will also use his term, as it is more precise than is "real": for instance, as he argued, idealism (a vastly different position) shares the same premise, "that what is known is antecedent to the mental act of observation and inquiry" (1929, p. 23). (Note however that this use of "ontological" is non-standard, as he pointed to aspects of the concept that are not among those that are usually considered.)

Constraints as instruments

The alternative mode of explanation which the source of control principle adheres to is *pragmatism*, and it takes a fundamentally different starting point. In this view, a constraint is an *instrument*: that is, it is created for a purpose; by someone; and as a means to an end. And as an instrument it is actively *formed* to serve its purpose, by the person applying it toward this purpose.

As a consequence of this view, constraints are not fixed or static; they develop through the process which adapts them to their function. Also, they are not considered given, whatever that may exactly mean, or assumed to have an independent existence, instead they have to be created by someone, and this requires effort. And they are not objective, but neither are they arbitrary, because they have a *purpose*.

The source of control principle directly embodies the principles of pragmatism, even though Lawson (1980) who first described it made no such connection. The function of constraints as instruments is easiest to see when they are under the designer's control; then, she can freely elect and discard them, or change them. In this way, she can flexibly create a requirements specification that will serve its pur-

pose, that is, to yield a good design solution. This is what we saw in the case of the "distributed control system" constraint above.

From the same principle, it becomes clear why rigid constraints are problematic, and why this rigidity arises when they are not under the designer's influence: when the control over a constraint is gone, the negotiability goes with it, so when it cannot be made to fit its purpose, then its role as an instrument is lost as well. The remoteness/control dimension can then reconcile all the seemingly contradictory properties, and explain the widely different kinds of constraint, by regarding them as instruments. Compared to this, the traditional realist view can account for very little about how constraints work under authentic circumstances.

Pragmatism as such

I have compared realism and pragmatism as two theoretical perspectives on constraints and the problem itself. In its essence, pragmatism is a theory of knowledge, or in the term preferred by pragmatists, *knowing*.

Pragmatism was originally the position that the meaning of a concept lies in its practical (i.e. pragmatic) consequences, and was founded by Peirce and James (e.g. Peirce 1931, James 1907). As it matured, it grew into a comprehensive, fundamental reorientation in the view of knowledge. Many of the most important developments were due to the work of Dewey (e.g. 1903, 1929, 1938, 1949). My treatment will accordingly be based on his work.

Whereas previous theories of knowledge had been based on purely philosophical issues, pragmatism eventually became a comprehensive theory based on practical matters of knowledge.

The previous "ontological" view regarded knowledge as reflecting eternal universal facts and truths, with mathematical knowledge as prototypical. To pragmatism, knowledge has the purpose of serving an individual by giving her practical and adaptive advantages; this perspective was greatly influenced by Darwin's theory. Here the empirical sciences also replaced mathematics as the model and context for scientific knowledge; the developments of relativity theory and non-Euclidean geometries became important cases in point (Dewey 1929, 1938, 1949).

Pragmatism prefers the term "knowing" to knowledge. It is a label not for a thing but a capacity, something that manifests itself in an individual's actions and which is not assumed an existence beyond that:

knowing is thereby primarily an activity, and this is reflected in "knowing" being principally a verb; knowing is an entity only in a derived sense, and this is reflected in "knowing" also being a verb used as a noun, and not a noun *per se*. "Knowledge" is a noun, pointing out a thing stored in the mind.

The theory of inquiry

One major element in Dewey's contribution is his *theory of inquiry* (1938, 1949). The pragmatist view of knowledge was originally a somewhat abstract position, meant as an alternative to the equally abstract point of view in which knowledge consists of semi-linguistic, logical propositions that express ontological truths. It was a fairly general statement of knowledge being grounded in practical activity and use; although it would gradually become a more articulated position. Since pragmatism stated that knowing is grounded in use, an articulated position would also have to specify what "use" consists in. Dewey's theory of inquiry is such an articulation of the use and activity that knowing is part of.

In comparison, the previous view of knowledge was of a set of logical propositions about the physical world. That view of knowledge was also accompanied by a theory of cognition of sorts; of the processes in which that kind of knowledge is used, and put to use. This role was played by the (Fregean) theory of formal logic and deduction, i.e. the view of cognition as formal, abstract reasoning based on symbols, propositions and so on.

Since the prototypical kind of knowledge was mathematical, also the way in which such knowledge is used, i.e. in deduction, proofs and formal logic, became the paradigm for reasoning, and even the fundamental model of cognition in general. Compare this with Pappus' method being the model of mathematical reasoning and logic, as discussed in chapter 1, and it is plain to see why his method could have such thorough influence on cognitive theory.

By analogy, the theory of inquiry is the corresponding pragmatist theory of cognition. The notion of "inquiry" itself refers to those adaptive and practical, concrete activities where knowing is put to use. Cognition is held to consist in the *entire* activity of inquiry, not merely in a process of pure, abstract thinking. Also, *all* cognition consists in inquiry; it is the basic structure of cognition. (The best definitions are given in Dewey & Bentley 1949.) My use of the concept will only include the parts of Dewey's theory which are relevant for

the present purposes, and it is by no means intended to be complete. However, on the points that I do include, my account goes well beyond Dewey's original version.

As witnessed in the book title *Logic: the theory of inquiry* (1938), Dewey also intended his theory as an alternative to formal logic, both as a model of scientific inquiry and of cognition in general. One must remember, though, that pragmatism and inquiry do not simply make up a set of new and different answers to the same questions that the traditional theories of knowledge are concerned with. For example, pragmatism is concerned with the use and function of knowing, which is all but irrelevant to the ontological view; accordingly, ontology is not a concern for pragmatism.

The instrumental view of knowing, as always being a means toward an end, is the innermost essence of pragmatism. Constraints, as discussed above, are one instance of this view. What it means to "treat knowing as an instrument" can be divided into two dimensions, one "logical" and one concerning process. The "logical" dimension concerns the attitude that is taken toward knowing, as having a function or purpose (versus having an independent existence and a preexisting and fixed meaning, apart from purpose and specific context of use). The process dimension concerns how knowing is "treated" in the most concrete sense: Firstly, how and when it is used for its purpose, the activities it is part of, and the specific actions that are taken there; And secondly, since it is not given, how it is created and adapted to its purpose. These processual aspects are what the theory of inquiry covers, and that I, instead of giving an account of Dewey's theory here, will introduce "as we go" in the remainder of this chapter: chiefly, these are the inquiring or cognitive function of action, the dual use/test purpose of action, and the developing dimension of inquiry.

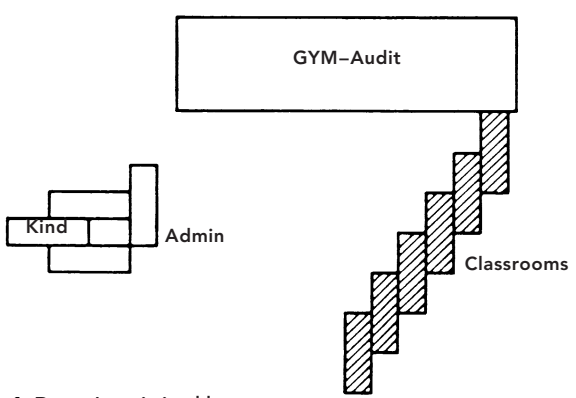


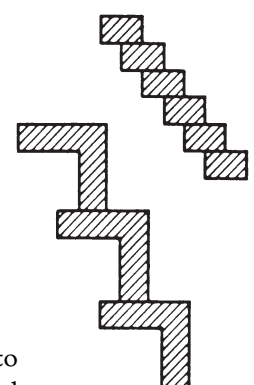
Figure 3.1 Petra's original layout.

3.4 Problem setting

I will now move on to how the problem is treated in design, and to look at a classic example of problem setting, which is Schön's own example, made famous from several of his writings (1983 ch. 3, also e.g. 1987, 1992). It describes a dialogue where an architectural student, Petra, reviews her work on a design project with her project supervisor, Quist. The review takes place at an early stage of her work.

The design review begins with Petra describing her work so far, and the problems she is having. A basic precept of architecture is that a building should be sensitive to the site it is located in, and that the physical, three-dimensional form therefore should fit into its surroundings and the location. Petra describes how she has taken this as her starting point, by trying to fit her design to a prominent land contour on the site. This is also where she has run into problems; she hasn't been able to fit the building into the slope. This issue has brought her work to a halt, and now she feels that she is seriously stuck:

Petra: I am having problems getting past the diagrammatic phase —I've written down the problems on this list.
I've tried to butt the shape of the building into the contours of the land there—but the shape doesn't fit into the slope.
[She has a three-dimensional model which she is referring to]
I chose the site because it would relate to the field there but the approach is here. So I decided the gym must be here—so I have the layout like this.
[She shows a rough layout, see figure 3.1.]



Quist: What other big problems?

P: I had six of these classroom units, but they were too small in scale to do much with. So I changed them to this much more significant layout [the L shapes]. It relates one to two, three to four, and five to six grades, which is more what I wanted to do educationally anyway. ...

Later, as Petra's problem description becomes clearer to Quist, he enters into her description with comments, and

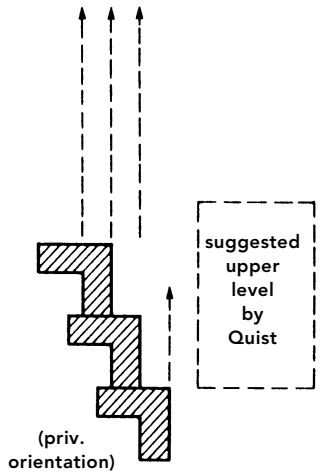
it turns into a dialogue that revolves around the drawings in front of them:

P: This is the road coming in here, and I figured the turning circle would be somewhere here—

Q: Now this would allow you one private orientation from here and it would generate geometry in this direction. It would be a parallel...

P: Yes, I'd thought of twenty feet...

Q: You should begin with a discipline, even if it is arbitrary, because the site is so screwy—you can always break it open later.



In this last statement, Quist begins to work out his diagnosis and remedy for Petra's stuckness. Schön's analysis (based on the events that follow later in the protocol), gives the following rationale for what Quist does (1983, p. 85):

> The main problem, in Quist's view, is not of fitting the shape of the building to the slope; the site is too "screwy" for that. Instead, coherence must be given to the site in he form of a geometry—a "discipline"—that can be imposed on it.

The reason for Petra's problems is that the site in question is not suitable for adapting the building to it, which is otherwise a basic principle in architecture. This is in Quist's view why she got stuck. Instead, Quist suggests an entirely different approach. Just *because* the site is "screwy", the designer should bring with her a certain structural principle to the site, as part of her design solution. He thereby doesn't say how to solve Petra's problem—how to fit the building into the slope—instead he proposes to *change* her problem into a different one. This is what is known as "reframing", or in Schön's words, "problem setting".

The coupling of problem setting and problem solving

Here, Quist has just demonstrated an instrumental attitude toward the problem itself. When he sees how Petra is stuck, he immediately changes the way she has set her problem. He thereby treats the framing of the problem as an instrument to adapt to the purpose at hand, here, to yield a good design solution.

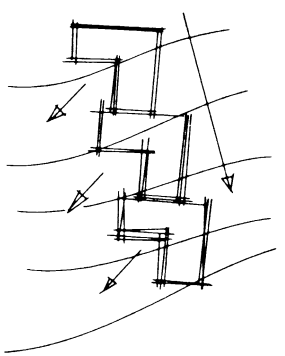
In the instrumental view, also the problem itself has a purpose. The problem statement serves to specify the function of the design solution. Hence, when the problem is not regarded as something given, it is also no longer a hindrance making life difficult for the designer. Instead, it serves to spell out the purpose of the eventual design. And the activity of problem setting becomes an inquiry into this purpose, in order to understand what it is. Thus also the task of problem setting makes a contribution to the designer's understanding.

This is seen more clearly in terms such as "requirements analysis" and "requirements specification", which correspond to "understanding the problem" in problem solving terminology. From these terms it is more obvious that this is where the function of the design is determined. Perhaps these terms are more informative because this role is more evident in design, where this activity hasn't been reduced to understanding a given problem by reading a piece of paper.

Solving is also use & implicit test of the problem-setting

When the protocol continues, Quist demonstrates the practical work that is required for successful problem setting. After having placed a transparent sheet of paper over Petra's sketches, he starts to draw over them:

Q: Now in this direction, that being the gully and that the hill, that could then be the bridge, which might generate an upper level which could drop down two ways.



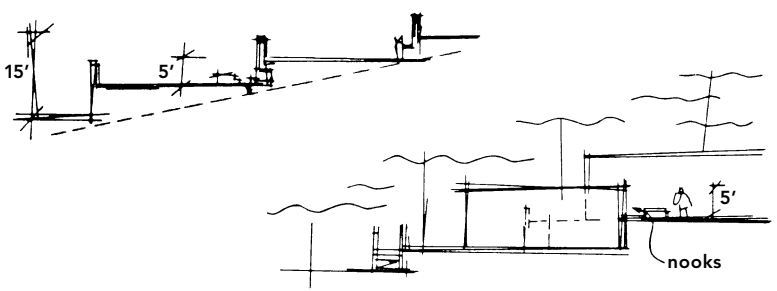
Here, Quist immediately starts to work out the implications of the new problem-setting that he has just suggested, which states that the function of the building's "external geometry" is to impose an order on the slope. He assigns this role to Petra's line of L-shaped buildings, placing them on the slope of the hill. If there is enough geometrical form inherent in this arrangement, then it will create the desired order on an otherwise "screwy" site.

Thus, when Quist starts to work out a solution, this also serves as a test of the problem-setting he has just proposed: If he can create the

order that it calls for, then this shows that he has set the problem properly. At least to a certain extent, as there are many other factors that the final design will have to satisfy. On the other hand, if he like Petra fails to produce a satisfactory solution, then he should probably change the framing again.

However, Quist does not perform any explicit test; instead, it is the act of working on a solution that also serves as an implicit test of the problem being solved. Hence, also the outcome of the test is implicit. In particular, success consists in the absence of failure, showing that the problem enables him to make progress:

> Q: We get a total differential potential here from one end of the classroom to the far end of the other. There is 15 feet max, right?—so we could have as much as 5-foot intervals, which for a kid is maximum height, right? The section through here could be one of nooks in here and the differentiation between the unit and this would be at two levels.



Thus, when Quist here moves swiftly from issue to issue, one idea generating another, then this easy progress should be seen as evidence of a problem well set—no news is good news. If there were problems, then we would see them in his work not moving so smoothly, but instead becoming erratic and staggering, bumping into obstacles and not moving forward. Instead, each "right?" works as one link in a chain of possibilities.

As Schön showed, from Quist's comments we have implicit evidence that he is attending to the feedback from his actions and the emerging sketch; of whether they tell him that the solution (and thereby the problem) is good or bad. After a few more moves, he acknowledges more explicitly that his drawing has served as a test:

> Q: The kindergarten might go over here—which might indicate that the administration over here—just sort of like what you have here—then this *works slightly* with the contours—then you might carry the gallery level through and look down into here—*which is nice*. (my italics)

The comment that the solution "works slightly with the contours" refers back to his original problem-setting, stating that the solution has been brought far enough along for him to deem it as workable and "nice"—if not an unqualified success, which it is still too early to judge anyhow. He had set out to fit the layout of the building to the screwy site, and he has now found this to work somewhat, and it thus seems a viable approach.

The inseparable use/test dimension of inquiry

The rationale for Quist's actions here is given by an essential element of all inquiry, which is that action has not one but two kinds of purpose or effect, in that it works as both use and testing at the same time. When action makes *use* of some piece of knowing, it is at the same time a *test* of that knowing. In the present case, by working on a solution Quist has been applying his problem-setting to its purpose, which is to yield a good solution. The solution he has worked on, and the act of working out this solution, have therefore also served to test whether his problem-setting works.

This dual purpose of action deviates from the ordinary, common sense view where action is associated with only one function, which is to produce a certain result. I will refer to this ordinary function as the first, *productive purpose* of action. As the other purpose serves the inquiry itself, I will refer to that as the second, *inquiring* or *cognitive purpose*.

Use & test, and the dual purposes, are not separate components that mix and together become part of an inquiry. Rather, they are different types of effect of the same single action or activity, facets which become visible by taking different perspectives or points of view. Either perspective may be applied to any action, and also on any level, including activities on a larger scale.

This also applies throughout the present episode of Quist drawing, from reframing Petra's problem to concluding that it works. From the use angle, the act of drawing serves to develop a solution, with the usual purpose of moving toward a final design. But from the test point of view, it can also be seen as an extensive evaluation

of Quist's problem-setting. This is what Schön (1983) refers to when he describes this passage as Quist performing a *frame experiment*. And each action that in the ordinary view serves to develop the solution fits the test interpretation equally well, serving the inquiry into the problem-setting. The two perspectives are thereby entirely complementary and symmetrical, neither being subordinate to the other.

Even though Quist's drawing episode thus serves both purposes, the productive outcome is of secondary importance in the greater context of his helping Petra; after all, Quist's job is not to produce Petra's solution, but to get her on her way again; the frame experiment serves to ascertain that his suggested remedy will be effective. The production of a solution thereby *primarily* serves the frame experiment, that is, the result may well be scrapped and still have fulfilled its purpose completely. Even the problem solving may be said to have a primarily inquiring purpose. This goes to show that the inquiring purpose of action isn't necessarily subordinate to the productive purpose, but potentially even the reverse.

Still, because of the very tight coupling between test and use, it is *necessary* to put the problem-setting to use in order to test it. For this reason, problem solving cannot be separated from problem setting, to be performed as separate activities, in separate phases of the design process. This is the single most important consequence for the present argument, as it shows why the separation made by Pappus, and carried on into contemporary theory, is simply impossible. It also shows that it is impossible even in principle, not merely for practical reasons or because of accidental circumstances.

The more general version of this argument is that all knowing must be tested by being put to use; this, in turn, so as to be adapted to its purpose. Hence developing understanding, i.e. "learning", cannot be separated from using that understanding for some purpose.

3.5 It's good to be a pragmatist

Because actions are acts of knowing, they test whether that knowing is able to serve its purpose or not, by either succeeding or failing. Hence, action has a second, inquiring purpose, and because the test is implicit, a positive outcome merely consists in the absence of failure.

Thus, there is no distinct evidence of success, nor even of a test having taken place at all. But what about a test that *fails*? In fact, nor in this case is there any explicit evidence. Instead, the action simply doesn't work: It either yields a different result or none at all; you try

different solutions and new angles, but nothing works; eventually you run out of ideas and find yourself stuck. In this manner, a failed test consists in not being able to move forward, but there is no overt signal of failure.

In fact, it was such a situation that developed when Petra got stuck, and which she described to Quist. She had tried to fit the building to the site but without success, and she couldn't "get past" this problem—she had even concluded that her own problem-setting was undoable, as fitting building to slope was impossible; however, she obviously didn't realize this:

P: I am having problems getting past the diagrammatic phase—I've written down the problems on this list.
I've tried to butt the shape of the building into the contours of the land there—but the shape doesn't fit into the slope.

Petra acts as a realist

However, Petra's own account does not imply a failure; it is evident that she has not seen her own actions as a test, neither when this occurred, nor in her after-the-fact description to Quist.

But Petra's view is in no way in error, as there are no overt (ontological) grounds for the "action as test" interpretation, neither with respect to the outcome of the test, whether it be a success or a failure, nor even of there being a test in the first place (since the test is implicit). There is no objective circumstance that she has missed or left out.

Still, her view has consequences that cannot be denied, since her problems are due to her not seeing the test dimension: Thereby, she doesn't listen to the feedback from her own actions, which would have told her that her framing was problematic. Neither does she view the negative outcome as a *failed test*, only as a *failed solution*; and so the only remedy she sees is to *change the solution*. When she restricts her efforts to trying new solutions, she has thereby locked onto one particular problem framing, as though it were the correct one, or the only one possible.

Hence she doesn't see the problem as being testable and thereby potentially better or worse, and as her own creation that is only one among many, and that she is free to change. She instead treats the problem as though it were given, and doesn't question its validity or utility; to Petra it is *the* problem, not *her* problem.

She thereby also doesn't see that her solutions are shaped by her framing, which reflects her understanding of what a solution must achieve, and which thereby also drives her solutions in a certain direction. She doesn't see the shortcomings of her attempted solutions as opportunities to understand better what a solution should do, i.e. to frame the problem better.

The bottom line here is that there is an unmistakable pattern in Petra's behavior: she consistently demonstrates the stance of a *realist*, and this is what causes her to get stuck. By not seeing her solving also as a test, she doesn't recognize the second, inquiring purpose of action. She thereby misses the signals that indicate a problem with her framing. And by neither seeing, nor using the option to *change* her problem-setting, she treats it as if it were given, instead of regarding it as an instrument and making use of it in what she is doing. It was her inability to see that she had set her problem badly, and failing to change it, that caused her to get stuck. These are all characteristics of the realist point of view, which had some very real, negative effects on her work.

Quist acts as a pragmatist

Quist, however, consistently acts as a *pragmatist*. He acknowledges Petra's solutions as being directed by her problem-setting, and thereby sees that the deficit lies not in the failed solutions themselves, but in her problematic framing: In a sense, all solutions were doomed to fail from how she had set her problem, as the impossible task of fitting the school to a screwy slope. Here, a critical element is Quist's ability to recognize the condition of being stuck; i.e. that further attempts at a solution are wasted, and to therefore switch from solving to setting the problem.

Quist also regards the problem-setting not as given, but as being created and shaped so as to serve its purpose, and as being useful when this is done right—but also that this requires that it *be made* useful. He also seems to know that he must test his framing to find out whether it works, by trying to solve it and thereby apply it to its purpose. Thus, he thoroughly acts as a pragmatist: he treats the problem setting not as given but as an instrument that you shape to make it serve its purpose. He also recognizes the inquiring function of Petra's and his own actions, and subjects his framing to an inquiry, and so on. Above all, this makes him successful where Petra got stuck.

Good designers are pragmatists, novices are realists

Earlier I compared the realist and pragmatist views of constraints, then as two approaches to scientific explanation. What we have seen here is these attitudes being taken by the designers, not by scientist observers, and not merely as ideology, but in their concrete actions.

But also Petra's description of her problems is a realist's account of her situation, very similar to the ontological view of constraints seen earlier, neither of which is very informative: the whole pragmatist dimension is missing from her analysis, which therefore captures very little of what had happened. It merely states that she had repeatedly failed to come up with a solution to her problem. Above all, it gives no clues to her available options for action, for how to resolve the situation and move on.

Hence, the two contrasting attitudes make the whole difference between frustration and progress: Quist literally *makes* his problem solvable, whereas Petra *finds* herself stuck. The bottom line is that Quist who is the "expert" is acting as a pragmatist, whereas Petra, the "novice", acts as a realist. And as we have seen, this accounts for a great deal of his superior performance. The choice of either position is not merely a matter of ideology, but has important consequences. For pragmatists, very practical consequences.

This expert–novice difference is further supported by Lawson:

> Students of design often devote too much of their time to unimportant parts of the problem. It is easy for the inexperienced to generate almost impossible practical problems by slavishly following ill-conceived formal ideas which remain unquestioned but could quite easily be modified. One of the major roles of design tutors is to move their students around from one part of the problem to another and the job of the design student is to learn to do it for himself. (1980, p. 81)

Here, students "generating impossible problems which remain unquestioned" is precisely the failure to act pragmatically, and "move the student around" was exactly what Quist did to help Petra. Lawson also makes the same point regarding constraints: when students impose their own constraints, they do not realize that these indeed are of their own making and not something given or "found":

> It is obvious that these designer-generated constraints are comparatively flexible. If they cause too many difficulties, or just

> simply do not work out the designer is free to modify or scrap them altogether. Design students often fail to recognize this simple fact but instead continue to pit their wits endlessly and fruitlessly against insuperable problems which are largely of their own making. One of the most important skills a designer must acquire is the ability critically to evaluate his own self-imposed constraints... (pp. 70–71)

Lawson regards the pragmatist stance as "one of the most important skills" to learn in becoming a proficient designer, one who knows how to put her instruments to work. Hence, in this view design education consists in turning realist students into pragmatist designers.

Expert–novice theories and cognitive science

The expert–novice dichotomy has been very popular in cognitive science (e.g. Chi, Glaser & Farr 1988, Ericsson & Smith 1991). An early theory of expert–novice differences was that of Newell & Simon. In accordance with their theory of problem solving, they proposed that the leg up that experts have is superior general problem solving strategies, namely those embodied in the authors' theory of problem solving; means–ends analysis and so forth. However, it turned out that the very opposite was the case (Holyoak 1991): experts have domain-*specific* skills that give them their advantages. *Novices*, on the other hand, when they have nothing else to go on, fall back on these most general techniques, means–ends analysis and so forth, as their last resorts in lack of other alternatives. And these techniques turned out to be the weakest of all problem solving strategies (*ibid.*).

What we have seen here is a similar case. The performance during laboratory problem solving (with a fixed, indeed *given* problem-setting, etc.) reflects how novices work on design problems, and yields clearly inferior results for them. Expert performance under realistic conditions is quite different from what is observed during laboratory studies.

3.6 The developing dimension of inquiry

One might say, taking a step back, that Petra's failure in a broader view is that she fails to see the need for *making* her problem-setting useful. She seems to think that once she has found a problem—*the* problem it seems—then that's it. Instead, problem setting is a process where the problem is evaluated and modified if necessary, so as to adapt it to its purpose.

Petra does however recognize the need for making the *solution* useful, and she performs the necessary actions of an inquiry: her initial idea was to line up the individual classroom units in a diagonal row. She developed this idea on paper, and responded to the feedback received by forming a line of three L-units that added the missing qualities.

By doing this, and by moving on to further issues as each problem is addressed, a solution will come to develop. Eventually, when the individual actions can be seen as parts of a greater whole, then they cluster into patterns where the knowing is incrementally adapted to its purpose through the individual actions, even including the false starts and having to back up from dead ends. This is the *developing dimension* of inquiry.

So whereas Petra does perform all the necessary steps of an inquiry in order to develop the *solution*, she fails to do the same with the *problem*. She appears not to see the need for doing the *work* of producing a good problem-setting, and that this is her own responsibility.

Quist recognizes precisely this need for *making* the problem solvable: problem setting is not only the act of proposing a new framing, but the whole process whereby you test it and refine it, so as to make it useful. Petra also *proposes* a problem-setting, but she fails to subject it to inquiry as Quist does: he begins to work out a solution to test the problem-setting, but also to understand the problem better, to *make* his framing useful.

He also states that his initial framing is merely tentative: "you should begin with a discipline, even if it is arbitrary, ... you can always break it open later". The initial form or quality is not critical. Since the inquiry is to ensure that the framing will eventually be useful, it can even be "arbitrary" at first, as long as it is enough to get the inquiry going.

The point of view of this as an inquiry thereby shifts the emphasis from the initial proposal, whose importance is played down considerably, to the *eventual outcome*; the product of inquiry.

The interplay between problem and solution

But because of the use–test duality, there is a reciprocal relation between developing the problem and solution, so that working on ei-

ther also serves to develop your understanding of the other, and vice versa. Therefore problem and solution are intimately connected and develop in parallel. This is why actual design work does not separate these two aspects from each other: on the one hand, it has proven impossible to separate them; but on the other hand, dealing with them together yields important advantages.

This can be illustrated by a study done by Nardi and others (1991, 1993), of an activity located in the border zone between design and small-scale problem solving, the activity of developing computer spreadsheet models. The first example is of an accountant who learned to develop such models himself, instead of having the company's programmers do it for him, as had been originally intended. The main reason why he did so lies in the close coupling between problem setting and problem solving. He found it impossible to describe to a programmer just what it was that he wanted. Instead, when he built the models himself, he could use the spreadsheets to develop his own understanding of what he wanted by working on the problem. That is, he wasn't able to formulate a problem statement without working on a solution to it—while doing so in contrast helped him considerably (1993, p. 13):

> *Jeremy*: We had to have rather large complex spreadsheets [for the business plans] where you had lots of variables. *And I found it easier to develop that myself than to go to somebody and say here's what I want, here's what I want, here's what I want*. And that's what really got me going on [spreadsheets] ...
>
> *Interviewer*: Why was it easier for you to do this yourself than to specify it for a programmer?
>
> J: I think it was easier because I felt that I was learning as I went, *as I was developing the spreadsheets, I was learning about all the variables that I needed to think about*. It was [as] much a prop for myself as [a way of] ... getting the outcome ... And there were a lot of false endings, I should say, not false starts. I'd get to the end and think, "I'm done," and I'd look at it and I'd say, "No, I'm not, because I've forgotten about one thing or the other."

First of all, the accountant states that he did not have a clear picture of what the problem was, even to himself, and much less one that he could give to the programmer. It seems paradoxical, but in the beginning he appeared to have a clearer picture of the *solution* he wanted than of the problem. This goes completely counter to the conventional view, but it makes sense in the way that he describes it. He couldn't make use of a programmer because it would require that problem setting *could* be separated from solving; not only between different phases, but between different *people*, even. It would have required that stage models worked as intended.

Instead he needed to develop a solution to understand what he wanted; spreadsheets enabled him to do this, and that was why he liked them. And when he describes his work on the spreadsheets as a "prop" for himself, he is referring directly to the second, *inquiring* function of this work, in that it also serves as a prop for his own thinking (about the problem), i.e. the inquiry, "as much as a way of getting the outcome". This last phrase is also a clear reference to the first, ordinary purpose of his work, that of producing the spreadsheet model itself. The way in which these spreadsheets are used parallels Quist's use of sketching to articulate his problem-setting, on point by point.

Lastly, he describes the dialectical structure of this work, which seamlessly shifts back and forth between problem and solution. It is clear that understanding the problem helps in solving it, but here the reverse is also obvious: a solution is not the end of the process, but only a false ending, as new solutions repeatedly serve to make him discover new aspects of the problem. Thus both develop in parallel, each as a "prop" for the other.

> In the study we found that spreadsheet users are very aware of the fact that their initial problem formulations are likely to be fuzzy, incomplete and badly structured. They like spreadsheet software because it helps them to work through these difficulties. (*ibid.*, p. 11)

Similarly, in a study of architects, as reported by Lawson (1980), Eastman (1970) showed "how the designers explored the problem through a series of attempts to create solutions", and found "no meaningful division" between analysis and synthesis, but rather "a simultaneous learning about the nature of the problem and the range of possible solutions". The designers "discovered much more about the problem as they critically evaluated their own solutions" (*ibid.*). This is a well-known phenomenon in design work.

In the second example, another accountant describes how she developed spreadsheets for an executive in her firm, and how the spread-

sheets worked as props for her boss. Her story seems to be taken right out of a "Dilbert" cartoon:

> Oh, this [spreadsheet] is what I gave to the CFO at first just comparing Q2 [Quarter 2] year-to-date budget to Q2 year-to-date actuals. And he said, "Well, for the board meeting I want [some other things]." Every time you do this he wants it differently. So I can't anticipate it. I just give him what I think [he wants] and then he says, "Ah, no, well, I want to have projected Q3 and projected Q4, and then total projected, and then the whole year's plan on there." (Nardi & Zarmer 1993, p. 14)

By merely seeing the model, the chief financial officer can better describe what he wants. Again, the early versions mainly serve to inquire into what he wants; hard work on these is probably wasted ("Every time you do this he wants it differently"). They should merely set in motion the process which will ensure the final quality. Progress is incremental, with new problems and solutions alternating as the successive frames in a comic strip, the untiring accountant walking to the chief's office and back again. It is also evident that the task of problem setting spans the whole process, the CFO not knowing what he wants till it is on his desk. This example also demonstrates what the accountant in the previous example gained, by doing both parts of the work himself.

3.7 No pure analysis

We can now return to the question of why design methods don't work. The answer supplied by the theory of inquiry is that there can be *no pure analysis*: the duty that has been assigned to the analysis phase cannot be performed by analysis alone; it needs to be performed together with the other activities of the design process: understanding the problem, working on solutions, and evaluating your work. Compare this with the following answer to the question of what was learned from the failure of design methods (Rittel 1972):

> ...that the design process is not considered to be a sequence of activities that are pretty well defined and that are carried through one after the other like "understand the problem, collect information, analyze information, synthesize, decide", and so on; and another being the insight that you cannot understand the problem without having a concept of the solution in mind; and that you cannot gather information meaningfully unless you have understood the problem but that you cannot understand the problem without information about it...

Here, the sequence of activities is stated as the basic failure, but as I have shown in chapter 1, it is not primarily the imposed order among the activities that is the culprit, as has been previously thought, but rather the separation between them, and from which also the ordering follows. This also includes Rittel's second point, the impossibility of keeping problem and solution apart, either logically or as separate processes.

As Swartout and Balzer state, this separation is a fundamental element of the rational models of action (as seen in the Rosetta stone, figure 1.15, specification and implementation correspond to analysis and synthesis):

> For several years we and others have been carefully pointing out how important it is to separate specification from implementation. In this view, one first completely specifies a system in a formal language at a high level of abstraction in an implementation-free manner. Then, as a separate phase, the implementation issues are considered and a program realizing the specification is produced. ... *all current software methodologies have adopted a common model that separates specification from implementation*. ...
>
> Unfortunately, this model is overly naive, and does not match reality. Specification and implementation are, in fact, *intimately intertwined*... (1982, p. 438, my italics)

Again, the clash between the ideal view of how things ought to be and how they actually are. Also Guindon has documented a study of software engineers in a series of papers (1987, 1988, 1989, 1990 a,b, 1992), showing in detail that the design process does not follow the pattern of "structured design methods" such as the waterfall model, but that it instead follows an "opportunistic" pattern that deviates from the norm. But she also showed that this was not a "failure" to use these methods, but instead that there is good reason not to do so (esp. 1990a, also cf. Hayes-Roth & Hayes-Roth 1979).

4 phases vs. 4 aspects of inquiry

The full schema of the rational models of action contain three major phases besides analysis: understanding the problem, synthesis/action, and evaluation. The theory of inquiry also contains three basic as-

pects: the two elements of the use–test pair, and the dimension of developing knowing. Note the close parallels between the elements of each model: between use and action, test and evaluation, and between developing knowing and understanding the problem.

The two in the last pair both address the need for "learning", roughly, what you need to know to solve the problem—although they take entirely different approaches to how this is done. The other two pairs are also parallel in their function, along with analysis/thinking. (In case there is any doubt, e.g. problem solving theory explicitly states that problem solving is performed by this part alone, see Newell & Simon 1972, Simon 1981.) While design methods place them after each other and in separate phases, according to the theory of inquiry they are inseparable. They are not even distinct parts but only different points of view that can be taken; potentially even of the same, single action.

Hence, the claim that there can be no pure analysis means that if the analysis is separated from *either* of the three other functions, it cannot do the work assigned to it; the theory of inquiry also explains why this is impossible, for each of the three "auxiliary" functions.

The main point with the discussion of problem setting was that the problem cannot be determined or fixated before the process of solving it begins, but instead that understanding and defining the problem amounts to a major part of the whole problem solving task.

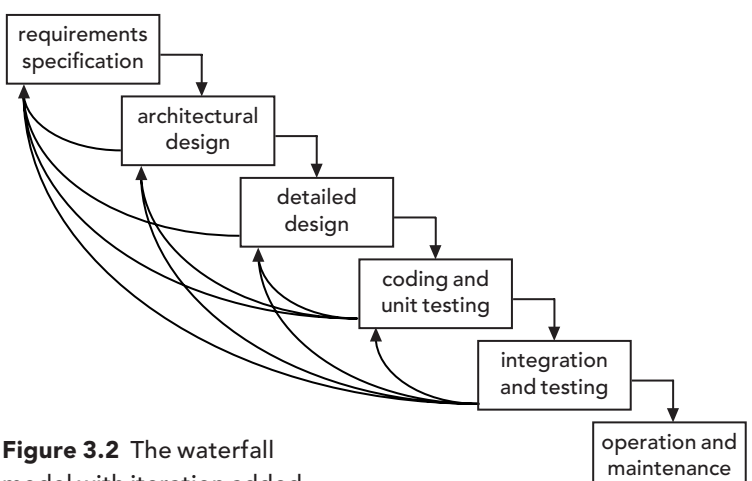


Figure 3.2 The waterfall model with iteration added.

And Nardi's examples, among others, showed that separating the two is undoable. The result of this is that setting and solving proceed in parallel, intimately intertwined and throughout the whole process, so that problem statement and solution are completed at the same time.

The simultaneous use and testing of your developing knowing, which is essential to working on problem and solution, complete the picture by showing how also action/synthesis and evaluation are inseparable from the other two components. Hence all the four elements of inquiry are tied together in a very fundamental manner. This is also corroborated by Parnas, who stated that both specification and modular decomposition (i.e. analysis) rely on an implementation process having already been done (Parnas 1985, quoted in Budde *et al.* 1992, my italics):

> There is, however, no method available for ensuring that a specification is complete and correct. *Only if a similar system has already been built* can a further system be consistently specified in advance with any amount of certainty.
>
> ...The decomposition of the overall system into modules is a simple matter in cases where the design decisions arising during implementation of these modules are known. This can only be the case, though, *if a similar implementation process has already been successfully carried out*.

That is, the prescribed procedures only work if you already have the answer, because you have already done the work once before.

Iteration

Someone might in defense of phase models refer to *iteration* as a well documented phenomenon (e.g. Carroll, Thomas & Malhotra 1979, Malhotra *et al.* 1980, Thomas & Carroll 1979). Adding iteration to a model means that you allow for the included phases to be repeated; this is represented by adding backward arrows to a box diagram (cf. figure 3.2, and also figure 1.16). However, on closer consideration, this supports the position I advocate. The reason is that iteration is a prototypical *ad hoc* extension, that is, an ill-considered added feature that handles a certain condition, but which in doing so goes against the original idea, and is therefore incompatible with it—thereby, in reality it constitutes no solution at all.

By allowing for iteration, a stage model comes to saying that you

can do anything, in any order, as many or as few times as you like. By allowing for everything, it no longer says anything about their order. But if you do that, you have given up what was the purpose of these models in the first place: to specify what things to do, when to do them, and in what order, so as to guide the designer. The only substance that remains is a list of the activities that are included.

And if a design method is such a list only, then you arrive at what I have stated—that design consists of several component functions that cannot be held apart, and that display no general ordering principle among them. Hence, the idea of iteration goes to say that the original idea of separated and ordered activities does not hold. In severe cases, an *ad hoc* extension can even divest the basic model of its original purpose, and this is what iteration does.

The cognitive roles of the 3 "auxiliary" activities

As a result, the roles assigned to the three "auxiliary" activities—evaluation, action/synthesis, and understanding—differ sharply between the stage models and inquiry theory. In the stage models, these are all downgraded to mechanical, empty, and meaningless activities, with the result that all the important work is pushed back inside the analysis. The most obvious example is the final evaluation stage, which can make no contribution at all to the design, as it is performed only when the design has already been completed. At most, evaluation is held valuable for *future* designs, which may draw on the experiences gained from the evaluation. Just such a role is also given to evaluation in problem solving by Polya (1945).

In inquiry theory, in contrast, evaluation has a quite crucial role, leveraging each successive attempt by enabling it to draw on the experience from previous trials; experience which evaluation generates by drawing out consequences and lessons learned from these attempts. It *can* have this role, as it is performed concurrently with the other functions. The role is directly reflected in the notion of *formative evaluation*, i.e. of the kind that serves the formative stages of design; such practice has increasingly been lifted forward as an invaluable but undervalued design technique (e.g. Carroll 1997, Hix & Hartson 1993).

Similarly, action/synthesis is an entirely mechanical phase which can have no secondary, inquiring function, since it via the plan is completely predetermined by the analysis/planning phase. And since

it is done only after the analysis has been completed, it cannot have any inquiring function.

And finally, understanding the problem is reduced to reading a given problem statement; compare with the previous statements of this as the hardest, most important and most difficult work of real design. This is also why phase models require that all information be *given*; pure perception or "input" is not capable of *generating* all this required information by itself; that takes *inquiry*—however, if it were given, reading alone might suffice. Compare with the Parnas quote (p. 98) stating that a complete specification is impossible to produce with less than having previously made a full implementation.

The need for intramental magic

Having trivialized these three other functions, the separated models push back all the important work into the analysis part; the intramental "black box" thereby seems to require magical powers when the whole task of design is assigned to it alone; hence, the air of mystery surrounding the concept of "creativity". In the inquiry model, the other three functions can make important contributions, and the burden on the mental part, and the need for magic, diminish accordingly. It seems far less puzzling that an architect can come up with an elegant, creative solution by diligently working at his drawing board, if she were held to do this just by having a creative idea, or sitting down to "think out" the same thing.

There is a saying that genius is 1% inspiration and 99% transpiration. If this is compared to the work of inquiry, the ninety-nine percent of non-intramental transpiration seem much less a waste of time on false leads, than an essential part of the work involved; the remaining percent then appears less wizardly. In fact, Thomas Alva Edison, who is accredited with this saying, apparently made hundreds of discarded "solution attempts" before building the one that would become the light bulb. We all know that we use less than ten percent of our brains' capacity; still, to claim that the non-mental parts stand for 99% of our cognitive capacity would be to go just *slightly* too far.

Cognition is inquiry, not intramental thinking

In summary, the roles of the three other components of inquiry are the main reason why the separated models are so weak; they fail because the unit doing the cognition has been deprived of the important services *to cognition* that these other functions provide. Without

these contributions, an isolated, intramental, "pure" analysis is made powerless.

Hence, not merely analysis produces the proof/plan/solution by itself, as the rational action models state, but all the four components of inquiry together are required for doing this; whether it concerns geometrical proofs, design, or other problem solving. And this is my general point about cognition, too: it does not consist of "pure thinking", but of *inquiry*, including all the four aspects I have enumerated here.

And now, having so far devoted an unproportionate amount of attention to the mental aspect, the rest of the book will be devoted to the cognitive roles of the other, shall we say, 98%?