ELSEVIER

Research Policy 24 (1995) 1-12

research policy

How learning by doing is done: problem identification in novel process equipment *

Eric von Hippel and Marcie J. Tyre

Massachusetts Institute of Technology, Alfred P. Sloan School of Management, 50 Memorial Drive, E52-556, Cambridge, MA 02139, USA

Final version received July 1993

Abstract

The unit cost of producing manufactured goods has been shown to decline significantly as more are produced. It has been argued that 'learning by doing' is at the root of this phenomenon, but the modes of learning actually involved have not been studied in detail. In this paper we attempt to provide a better understanding of the learning behaviors involved in learning by doing via a study of 27 problems that affected two novel process machines in their first years of use in production.

First, 'interference finding,' is described, a form of learning by doing that appears to be central to the discovery of the problems studied. Next, the reasons why the problems identified by templating were not discovered *prior* to field use - before 'doing' - are explored. Two causes are identified: an inability to identify existing problem-related information in the midst of complexity, and the introduction of new problem-related information by users and other problem solvers who learn by doing *after* field introduction of the machine. We find that problems due to information lost in complexity emerge earlier than do problems due to user learning by doing. Tests of reason are used to show why it would be very difficult to eliminate doing from learning by doing. Finally, other implications of the study findings are discussed.

1. Introduction

Beginning with Wright [28] a number of studies have shown that the unit cost of producing manufactured goods tends to decline significantly as more are produced. It has been argued that this effect is the result of the development of increasing skill in production attained by what Arrow [4] has termed 'learning by doing.' More recently, Rosenberg [20] has shown that similar gains can accrue to the end users of a product as their skill or understanding grows through 'learning by using.' (For example, after a given jet engine has been in use for a decade, the cost of maintenance may have declined to only 30% of the initial level as a result of learning by using [20, p. 131].)

Although the economic significance of learning by doing and using has been made clear, the *process* by which these gains are achieved is still quite unclear [1]. That is, we do not know the micro-level mechanisms by which learning by do-

* We would like to express our gratitude to Anne Carter, Shmuel Ellis, Dietmar Harhoff, Stephan Schrader and Stefan Thomke for their very helpful comments on the ideas embodied in this paper.

0048-7333/95/$09.50 © 1995 Elsevier Science B.V. All rights reserved
SSDI 0048-7333(93)00747-H

ing is actually done, nor do we know whether or why doing is essential to such learning. In this paper we explore these matters by means of an empirical study of a particular kind of doing, the identification and diagnosis of problems that affect novel process machines during factory use.

We begin (section 2) by discussing how learning by doing fits into the broader framework of problem solving. Next, we describe our study methods (section 3) and present our empirical findings (section 4). We then consider whether there is a substitute for learning by doing that does not involve doing (section 5). Finally, we discuss the implications of our findings (section 6).

2. Learning by doing as problem solving

Learning by doing is a form of problem solving that involves application of a production process in a use environment. In order to better understand this learning process, a brief digression into the general nature of problem solving will be useful.

Research into problem solving shows it to consist of trial and error, directed by some amount of insight as to the direction in which a solution might lie [6, pp. 43-47]. This finding is supported by empirical studies of problem solving in the specific arena of product and process development [3,15]. Such studies show trial and error (or, more precisely, trial, failure, learning, revision and re-trial) as a prominent feature.

Trial and error procedures guarantee a problem solution only in the instance of 'well-structured' problems, which are defined as those for which one can precisely *specify* a process of trial and error that will lead to a desired solution in a practical amount of time [18,19,21]. For example, a traveling salesman problem can be well structured, because one can precisely specify a generator of alternative solutions and a solution testing procedure that are guaranteed to eventually identify the best solution. However,

> In general, the problems presented to problem solvers by the world are best regarded as ill structured problems. They become well structured problems only in the process of being prepared for the problem solvers. It is not exaggerating much to say that there are no well structured problems, only ill structured problems that have been formalized for problem solvers... [21, p. 186]

Ill-structured problems may involve an unknown 'solution space' (a precisely specifiable domain(s) in which the solution is known to lie). They may also involve unknown or uncertain alternative solution pathways, inexact or unknown connections between means and ends and/or other difficulties. Ill-structured problems are solved by a process of first generating one or more (typically several) alternative solutions. These may or may not be the best possible solutions, one has no way of knowing. These alternatives are then tested against a whole array of requirements and constraints [15,22, p. 149]. Test outcomes are used to revise and refine the solutions under development, and, generally, progress is made in this way towards an acceptable result.

In sum, learning by doing and learning by using almost always address ill-structured problems. We can therefore anticipate that the problem-solving process associated with such learning will have the general characteristics described for ill-structured problems, plus some more particular attributes associated with doing or using.

3. Study methods

Our empirical study explored learning by doing associated with the use of novel process machines in factory production processes. We elected to focus our study on the early field use of two types of process machine, a solder paste profiler and a component placer. These machines were developed to automate manual procedures previously used to attach surface-mounted integrated circuits to large, complex circuit boards. [1] Both machines were developed by a computer manufacturer for use in its own factories. Although carried out within a single firm, the development of each machine was an independent event. Each was designed and built by different equipment development groups, and each was first applied in a different factory of the firm by different process engineers and plant-based users.

At the time of the study, both machine types had been installed in several factories. The first solder paste profiler had been placed into service 18 months before data collection began, and the first component placer 2 years before data collection began. As is often the case when novel machines are first introduced to the field, machine users had encountered a number of problems with the machine during this period of early use [11,14,24]. These field problems were the subject of our study of learning by doing.

The sample of field problems used as a study sample consisted of all machine problems that met the following three criteria: They had been observed after the machines had been introduced to the field; they were considered sufficiently serious to merit repair; they had been diagnosed as to cause (although not always fixed) at the time of the study. Problems meeting these criteria were identified by contacting both the engineers involved in using each type of machine in the factories where it had been installed, and contacting the engineers involved in designing the novel equipment. We asked each for an exhaustive list of all 'significant' problems observed after factory use of each machine began that had subsequently been diagnosed. (Note that under this procedure factory machine users determine both what constitutes a problem and what constitutes a solution. Thus, a problem can entail a machine failure to perform as designed, *or* significant user dissatisfaction with a machine that is functioning as intended by its designers.) A total of 27 problems were identified by this method, 12 affecting the profiler and 15 affecting the component placer.

Our analysis of patterns in learning by doing was based on a grounded research approach [8]. Data for our analyses were collected through interviews with both the users of each machine (the process engineers at the factories where they were installed) and developers (key people in the process equipment development teams). Most interviewees had been with the projects studied from their inception to the present. Initial interviews were conducted on-site where respondents could refer to contemporary logbooks and could demonstrate the problems they described on the actual equipment. Interviews lasted from three to six hours, including plant tours. Respondents were interviewed both separately and, to the extent possible, together. Follow-up questions were discussed in additional face-to-face meetings, and by telephone and electronic mail.

4. Findings: how learning by doing is done

In the course of our empirical study of learning by doing we explored three matters: (1) the link between factory use of a machine and the discovery of problems with machine functioning; (2) the nature of the problem-related information that was uncovered as a result of machine use; (3) the time at which different types of problems

[1] A brief description of solder paste profiling and component placing machines: The board assembly process begins with the application of a tiny dab of 'solder paste', a form of solder which has the consistency of toothpaste at room temperature, to each location on a circuit board where an electrical connection must be made between the board and an attached component. (The spacing between dabs can be as small as 25 thousands of an inch today, and each dab may be as small as the period at the end of this sentence.) Next is inspection (or profiling) of the solder paste, this is where the first machine that we studied (the 'paste profiler') plays its role. The machine scans the board surface with a laser-based vision system and determines whether the location, amount and configuration of each dab of solder paste applied to the board is as specified. If all is correct, the board is then passed on to the next operation, where components are placed on the boards.

The next machine we studied was designed to automate the placement of complex components. It uses a vision system and a robot arm to pick up integrated circuits (which look like small plastic boxes with two or more rows of tiny metal legs protruding from the bottom) and place them on the circuit board at precisely the right locations, with each metal leg of each component resting exactly on one of the dabs of solder paste previously applied to the board. When this step is completed, the board is passed through an oven that heats the solder paste and converts it into liquid solder. When the board cools, the solder hardens into solid metal and the 'placed' components have been permanently soldered onto the board.

were discovered. We report on each finding in turn.

4.1. Templating the process of problem discovery

A central form of doing in the factory is the use of process machinery in the course of production. We were interested in determining whether the problems in our sample were *first* discovered in the course of such 'doing', or whether some of these matters had been identified (but not fixed) at an earlier stage. We did this by asking the developers of the machines we studied to separate the sample of field problems we had collected into two categories: (1) problems the developers had first become aware of as a result of field use of the machines; (2) problems that they had known about prior to field introduction, but had not yet fixed due to constraints such as a lack of time. As can be seen in Table 1, 22 (81%) of the problems we studied were first identified in the course of field use. The distribution of our sample with respect to this matter shows no significant difference between the two machines studied ($p < 0.27$, Fisher exact test).

Thus, doing did appear to be closely associated with problem identification in the field. The 22 problems first identified during factory use were invariably first observed by factory personnel, who would report to the machine developers something like: "The machine stops working (or fails to perform as we want it to) under X conditions: Fix it!" Consider the following example drawn from our sample of cases.

Example: yellow circuit board problem

The component-placing machine uses a small vision system incorporating a TV camera to locate specific metalized patterns on the surface of each circuit board being processed. To function, the system must be able to 'see' these metalized patterns clearly against the background color of the board surface itself.

The vision system developed by the machine development group functioned properly in the lab when tested with sample boards from the user plant. However, when it was introduced into the factory, users found that it sometimes failed, and called this to the attention of the machine developers. The development engineers came to the field to investigate, and found that the failures were occurring when boards that were light yellow in color were being processed.

The fact that boards being processed *were* sometimes light yellow was a surprise to lab personnel. While factory personnel knew that the boards they processed varied in color, they had not volunteered the information to the lab because they did not know that the designers would be interested. Early in the machine development process, factory personnel had simply provided samples of boards used in the factory to the lab.

Table 1
When were problems affecting the machines first recognized?

	No. of problems affecting		
	profiler	placer	total
(1) *After* machine was installed in field, as a result of use – *Example*: After the component placing machine was installed in the field, users noticed that it was unable to pick up parts that had 'tilted' heat sinks on top. This problem was a surprise to developers. They had not known that such parts existed, and had not designed the machine to handle them.	9	13	22
(2) *Before* machine was first installed in field – *Example*: Specifications called for machine to handle all boards to be processed without needing extra setup. Developers couldn't find a way to do this during the development time frame; users and developers agreed that this problem would be resolved after machine introduction.	3	2	5
Total	12	15	27

And, as it happened, these samples were green in color. On the basis of the samples, developers had then (implicitly) assumed that all boards processed in the field were green. It had not occurred to them to ask users, "How much variation in board color do you generally experience?" Thus, they had designed the vision system to work successfully with boards that were green.

The yellow board problem illustrates recognition of an unanticipated problem as a consequence of doing, operating a machine in its actual use environment. But *how* does field use aid in problem discovery? The question is especially interesting because, given the additional complexity of using equipment in an actual factory environment rather than in a lab, one might expect that the difficulty of problem discovery would increase, not decrease with field introduction. In examining the process of problem discovery, we found a form of learning we call 'templating', a variant of trial and error problem solving, was present in all 22 of our cases of problem discovery through field use.

Templating can be described as a form of pattern recognition. A pattern is essentially a set of features or characteristics that describes an object (or event, or stimulus). This bundle of features then may be used as a standard against which one may compare new objects. Thus, one may wish to focus on the *similarities* between patterns in a process called pattern matching. (Systems designed to recognize objects with known characteristics ranging from handwriting to military targets often use algorithms based on pattern matching.) Or, one may wish to use subtractive pattern matching to highlight the *differences* between two or more patterns. For example, astronomers may compare two star maps of the same area of sky taken at two different times in order to 'subtract' everything that is the same and highlight only what is changing, rapidly moving comets for example.

Templating is a form of pattern matching which is sensitive to the interferences among objects (such as a process machine and a plant environment) that may have very different features or functions. Alexander [2, p. 19] describes the essence of templating when he discusses a means for characterizing the fit between form and context:

> It is common practice in engineering, if we wish to make a metal face perfectly smooth and level, to fit it against the surface of a standard steel block, which is level within finer limits than those we are aiming at, by inking the surface of this standard block and rubbing our metal face against the inked surface. If our metal face is not quite level, ink marks appear on it at those points which are higher than the rest. We grind away these high spots, and try to fit it against the block again. The face is level when it fits the block perfectly, so that there are no high spots which stand out any more.

The process of templating we observed in our sample of process machine problems is a more complex version of the process just described. Here, two very different and highly complex patterns, the new machine and the plant context, are brought in close juxtaposition during field use: 'doing.' As a result, previously unsuspected and often subtle interferences are discovered because they evoke an obvious symptom, poor machine performance. Thus, in the case of the yellow board problem described earlier, an obvious symptom (machine failure) led developers to discover that they had not properly adapted the machine to the color of circuit boards being processed in the plant.

In problem identification by doing, therefore, we find that the unique contribution of 'doing' to problem discovery in the field environment is precisely the *precipitation* of obvious symptoms. These are then traced via diagnosis [26] to previously unrecognized interferences between machine and use environment.

4.2. *Information availability and unanticipated problems*

We next focused on the 22 problems that were discovered as a result of field use, and attempted to understand *why* they had not been anticipated earlier. Since the causes of all of the problems in our sample had been diagnosed, we were able to approach this task knowing both the initial symptom and the 'cause' of each problem. (Problems can be understood and solved at many levels. For example, if machine operators find they must make frequent machine adjustments and find this troublesome, one level of solution would involve making the adjustment process easier. A solution at a deeper level would involve reducing or eliminating the need for adjustment. In our analyses we focused on the level of diagnosis and solution actually selected and implemented by the problem solvers studied.) We drew on the diagnosis of each problem to identify the information that would have allowed engineers to resolve each prior to field use, if only that information had been incorporated into the machine as originally designed.

We found (Table 2) that the information associated with a problem fell into two major classes with respect to its potential availability to machine developers during the design process. In 15 cases, the information existed in the use environment prior to and during the period that the machines were being designed, and so was potentially available to the machine developers for use in problem avoidance. In the remaining seven cases, the information that proved problematic was only introduced into the use environment *after* the machine had been designed and installed in the field.

To give the reader a better feeling for this distinction, and for the nature of the variability in information availability that we found, we will illustrate each of the four categories in Table 2 by means of a brief case example.

In cases tabulated under 1(a) in Table 2, the information needed to understand or predict problems did exist in the intended use environment during the development of the machine. Indeed, in each of the instances in this category, interviewees told us that the information could easily have been provided to the lab, had the developers thought to ask and/or had users thought to volunteer it. But, the relevance of the information was overlooked until it was made clear by templating during use of the machine in the field. The yellow circuit board case example presented earlier illustrates this category of problem.

In cases coded under 1(b) in Table 2, the information needed to understand or predict problems was actually present in the machine design [2] lab but, again, its relevance was not seen until made clear by field failure. This was often understandable: 'having all the information' did not mean that it was easy to predict the often subtle chain of cause and effect that eventually

Table 2
At the time the machine was designed, what was the availability of the information which could have been used to avoid an unanticipated field problem?

Availability of problem-related information	No. of problems affecting		
	profiler	placer	total
(1) Problem-related information *existed* in use environment when machine was designed, but:			
(a) was not known to machine designers	2	3	5
(b) was known but not used by designers	5	5	10
(2) Problem-related information was created *after* machine was introduced to field by problem solvers outside of the design lab who were:			
(a) users working directly with machine	1	4	5
(b) problem solvers working on other aspects of the production process	1	1	2
Total	9	13	22

resulted in an unanticipated field problem. Consider the following example.

Example: component slippage problem

Just before the component placing machine places components on a board, little dabs of solder-containing paste are applied to the board, one at each spot where an electrical connection is to be made between a component leg (a wire protruding from the base of the component) and the board. The machine designers knew about this, but chose to use adhesive tape instead of solder in their laboratory simulation of the use environment. (Use of solder would have required setting up the lab to comply with rules regarding the handling of hazardous materials, a costly matter.)

When the component placer was installed in the field, users noticed that components unexpectedly slipped sideways to an unacceptable degree when the robot arm was pressing them onto the board. Investigation showed that the mound-shaped dabs of solder paste were firm enough to push the component sideways if the legs touched down on their sides instead of directly on their tops. This effect did not occur in the lab because the lab had not used solder in its tests.

In the second category of Table 2, the information that might have allowed designers to anticipate and forestall a field problem was introduced to the use environment *after* field introduction of the machine by problem solvers who were not machine developers. In most instances (category 2a) these problem solvers were machine users who, in the course of their field experience with the machine, decided that they wanted something different from the originally specified performance. In many of these cases users experimented with changes to the use environment and/or to the machine itself [27] in order to develop their suggested improvements. Consider the following example.

Example: location adjustment problem

Each time a new board design was processed by the component placing machine, operators had to tell the machine where to put each of the components to be placed on the new board. They did this by entering the X and Y coordinates of each part location in the machine's computer memory. In case these coordinates required later adjustment, operators and machine designers both assumed that the operators would re-enter new X and Y coordinates.

After the machine was installed in the plant, users discovered that they had to adjust X and Y coordinates very frequently. They also found that it was very cumbersome to do this by re-entering new coordinates. Instead, they learned to make the needed adjustments via an obscure 'move it over by X amount' command that was buried several layers down in a software menu on the machine's control panel. The problem that users then brought to the attention of machine designers was: The 'move it over by X amount command' is very hard to reach and use. Make a more convenient one!

In two instances (category 2b), the problem solvers who created field problems after the machine was introduced were not machine users, they were individuals working on other aspects of the printed circuit board production process. Consider the following example.

Example: solder mask problem

Some months after the solder paste profiling machine was introduced to the field, engineers working on the printed circuit board production process decided to slightly reduce the thickness

[2] Three of the cases coded under 1b deserve special mention. In these, unanticipated field problems were caused by the premature failure of machine parts due to design error (for example, an inappropriately small bearing was designed into the machine, and quickly failed). It seems to us reasonable to classify these under 'information known by lab but not used' because the problems could have been anticipated and avoided prior to field use by using only information available to the lab and, for example, subjecting the machine to longer life tests in the lab. (The intended field operating life of the machine was known to the lab.) *If* the attributes of the use situation causing the failure had *not* been known to the lab in cases of premature parts failure (e.g. "We didn't know that you were going to process such heavy parts"), we would have coded the cases under category 1a in Table 2.

of the plastic film (called a solder mask) which served as the topmost coating of the printed circuit boards being processed. This was done to solve a problem unrelated to the profiler, the engineers wanted to improve the uniformity with which solder flux was being applied to the board. However, as an unanticipated side effect, the profiling machine's measurements suddenly became unreliable.

When engineers responsible for the profiling machine investigated the sudden rash of failures, they eventually found that the thinner solder mask was the cause. The profiler was designed to identify the top surface of the board to be measured by reflecting a laser beam from that surface. Introduction of the thinner solder mask resulted in greater amounts of laser light passing *through* the film and reflecting off layers of metal located inside the circuit board. As a consequence, the machine sometimes judged these lower layers to represent the surface of the solder mask film, which in turn led to incorrect measurements.

In sum, then, we see from Table 2 that some information associated with field problems existed at the time the machine was developed, while in other cases the information was created after the introduction of the machine, usually as a result of user learning associated with using the machine in the field.

4.3. Time sequence in problems identified by learning by doing

Next, we explored whether there was a difference in the timing of problem discovery when information pre-exists problem introduction, versus when information was created as a result of users' experience with the machine. It seemed reasonable that users would discover problems caused by pre-existing conditions sooner than they would develop new needs or encounter new conditions of use. As Table 3 shows, we do see a significant tendency in this direction. Problems due to pre-existing conditions were generally identified more quickly (within one month of machine introduction) than were those created by user learning or other changes in field conditions (Fisher exact test $P < 0.02$).

The pattern that we show in Table 3 cannot be taken as an iron rule. After all, desirable improvements might sometimes be perceived very quickly, and/or the symptom of an existing interference between a machine and a use environment might not occur immediately when the machine (or product or service) is introduced. With respect to the latter, consider that the machine and/or the environment might not be configured in a way that would cause a problem associated with a pre-existing field condition to be immediately expressed. (For example, if a problem was associated with the 'annual report' section of a software package, the user might not see a related symptom until that section of the package was activated.) Also, the symptom of a problem may not be manifested immediately, as in the case of premature wear failures in a machine. (We had three such cases in our problem sample, and two of these took many months to emerge.) None the less, it is interesting to see that the pattern shown in Table 3 emerges so clearly in this study sample.

Table 3
How soon after machine was introduced to the field was the problem symptom noticed?

Availability of problem-related information	No. of months after machine installed that problem symptom first noticed				
	$\leqq 1$	1-2	>2	n.a.	total
(1) Problem-related information *existed* in use environment when machine was designed	11	1	3	0	15
(2) Problem-related information was created *after* machine was introduced to field by problem solvers outside of the design lab [a]	0	2	2	1	5 [a]

[a] The sample in Table 3 is the same as in Table 2 except that the two cases in Table 2's category 2b are excluded from category 2 in Table 3. The reason: In these cases, the creation of problem-related information was independent of the machine under study. (Inclusion of these cases would have strengthened rather than weakened the statistical finding reported here.)

5. Does learning require doing?

We have seen that problems are discovered in the course of doing as a result of templating. In order to understand the need for 'doing' in this process, we next apply tests of reason to explore whether it would be possible to obtain the same learning without actually using process machines in their intended field environments. Rosenberg [20, p. 122] and Habermeier [10, pp. 276–278] have argued that doing or using is required because the possible interactions between products and their use environments are sometimes too complex to be predicted. In what follows, we offer support for this idea and develop it further. We distinguish between situations in which problem-related information is available at the start of a machine (or product or service) design project, and situations in which the problem-related information is only introduced after the machine is in use.

5.1. Learning without doing in stable use environments

As noted above, in many cases the information needed to predict field problems exists during the design process. Even in these cases, however, engineers who wish to predict all potential field problems face a difficult task, for two reasons. First, the use environment and the machine that will interact with it contain a myriad of highly specific attributes that could potentially interact to cause field problems. Second, which items among these will actually be associated with problems are *contingent* on the solution path taken by the engineer designing the product. We can illustrate both of these matters via the yellow circuit board problem described earlier.

With respect to the first point, note that the property of the board at issue in the yellow board case was problematic in a very narrow and specific way. That is, the problem with the board was not that it had 'physical properties,' nor that it had a color. The problem was precisely that the boards were yellow, and a particular shade of yellow at that. Since a circuit board, indeed, most components, have many attributes in addition to color (shape, size, weight, chemical composition, resonant frequency, dielectric constant, flexibility, and so on) it is likely that problem solvers seeking to avoid all field failures would have to analyze a very large (perhaps unfeasibly large) number of potentially problematic items and interactions to achieve this.

With respect to the second point, note that the problem caused by the yellow color of the board was *contingent* on the design solution to the component placing problem selected by the engineer, and this was only done during the development process. That is, the color of printed circuit boards in the user factory became relevant only when engineers, during the course of their development of the component placer, decided to use a vision system in the component-placing machine they were designing, and the fact that the boards were yellow only became relevant when the engineers chose a video camera and lighting that could not distinguish the metalized patterns on the board against a yellow background. Since engineers often change the alternatives they are developing during the course of their development work [3,15], the relevance of any particular item of information to potential field problems can also change frequently during the development process.

Of course, we do not intend to suggest by this litany of difficulties that one cannot anticipate and avoid a field failure when use environments are stable with respect to that problem's cause. It simply says that to do so can be complex and costly. Methods for reducing the likelihood of unanticipated field problems include simulating the use environment in the lab more completely: if the simulation is totally complete and accurate, one can cause all unanticipated failures to occur in the test lab instead of in the field. (This is the approach taken by airlines which seek to train pilots in simulators that are so accurate that simulator time is counted as the equivalent of actual flight time.). Also, one can use various analytical procedures such as 'fault trees' [12] which can help make the search for possible causes of failure more systematic. Further, one can hire very experienced engineers who have prior experience with failure modes on existing products, and so are more likely to anticipate them when designing similar new products. One can also try to incorporate subsystems in one's design which have already been tested under field conditions. Also, one can try to make some of the subtasks in a design project well structured so as to reduce the possibility of unanticipated field failure in these. [3] And, one can lessen the likelihood of failure by making the solution more robust, less dependent on possible variations in the use environment and/or more redundant. (The practice of incorporating safety margins into the design of bridges and buildings is an example of the first approach; the design of fault-tolerant computers an example of the latter.)

Both the costs and the benefits of identifying potential field failure prior to use of a new product differ from project to project. Learning by doing is the default strategy, other approaches are simply attempts to anticipate and prevent problems that will otherwise make themselves known through templating. Thus, one can expect that designers will invest more or less heavily in the fault anticipation strategies just listed depending upon the costs and benefits that they expect. For example, one would expect designers of nuclear power plants to invest a lot in attempting to anticipate and avoid potential field failures, and they do [17].

5.2. Learning without doing in changing use environments

The problems coded in the second category of Table 2 were *created* by changes in machine uses or the use environment that occurred after the machine was installed in the field. These changes were carried out by users (category 2a) or others associated with the production process (category 2b) rather than by machine designers.

The possibility that the use environment might change is a very significant matter to the designer who is attempting to anticipate and resolve potential field problems without 'doing.' When, as in the cases discussed just above, the designer is the only problem solver active on a problem, he or she is in the same position as a scientist or engineer asking a question of 'nature.' These problem solvers know that the answer they seek may be complex and hard to puzzle out. But they also know that it is not being changed as they work due to the actions of other problems solvers. For example, engineers building the first supersonic plane did not know all they needed to know about the stresses the airplane would encounter in supersonic flight. But they knew that nature would remain stable as they learned more, and that the correct answer would not change half-way through the project. In contrast, a use environment populated by and/or affected by autonomous problem solvers offers no such assurance. Under such conditions the use environment and thus the nature of the desirable solution that the designer is seeking to provide may well change during or after completion of the design process.

When problems are created by autonomous problem solvers, designers are *very* unlikely to be able to generate the same information by other means, thus avoiding related field failures and requests for improvement. The autonomous problem solvers are both posing hard-to-anticipate problems, and are generating an unpredictable set of proposed alternative solutions. Some of these may well involve changes in the machine (or product or service) provided by a particular manufacturer.

Neither game theorists' models of cooperative games ([5] nor psychologists' models of 'mutual adaptation' [13, p. 248]) offer us much help in predicting the path or the outcomes of this type of multi-party problem solving. Although both developer and user are presumably motivated towards mutual adaptation (or, at least, the ma-

[3] Despite the restrictiveness of the criteria for well-structured problems, designers can often partition an overall design task in such a way as to create some well-structured subproblems. For example, Smith and Eppinger ([23], plus private discussion with the authors) studied a subproblem in the design of automobile brakes that seems to us so tightly constrained as to meet the criteria of a well-structured problem. The goal was that 'the brakes on car model A should not squeal when they are used under test conditions X'. To achieve this goal, it was permissible to manipulate only three well-understood variables, such as the composition of the brake lining material, in precisely specified ways.

chine developers are motivated to adapt to their user-customers), the problems that machine users are framing and partially solving are, as noted earlier, ill structured. Therefore, as our section 2 discussion of ill-structured problems indicates, the problem solving path that will be taken by user problem solvers cannot be predicted by the designers with certainty.

6. Discussion

The approach we have taken to studying learning by doing involved conducting grounded research on multiple instances of a single type of learning by doing event, the identification of an unanticipated problem in a factory. We identified templating as a learning mechanism associated with this type of event. We also found that problems identified by learning by doing in our sample were associated with (1) information that existed in the use environment but was 'lost in complexity,' and (2) information that was newly introduced to the use environment.

We have observed templating only within a very specific context. None the less, it appears to be quite general, and may therefore be a useful way to describe the process of learning by doing and using [4] in a range of contexts. Templating may also prove to underlie a significant *proportion* of the gains associated with learning by doing. Thus, Mishina [16] analyzed the learning curves associated with the production of the B-17 airplane, and found that learning in production is more closely associated with changes to the production process than with the number of units produced over time. This finding is congruent with a central role for templating in learning by doing, because that mechanism applies specifically to adapting to novelty in the production process.

Our findings allow us to suggest a particular shape for a learning curve that will be induced by the introduction of a *particular* change into a use environment. Recall that we found that most pre-existing interferences between the new machines we studied and the use environment were flagged within one month of the machines' installation, while improvements derived from user machine-related learning followed later. If a significant proportion of the total problems flagged as worth working on were due to the identification and resolution of existing interferences, and if these were diligently diagnosed and solved – and they certainly would be if they caused grossly unacceptable performance – one would then find a relatively high rate of learning by doing immediately after the introduction of the novel element, that would drop to a lower level over time. A study by Tyre and Orlikowski [25] of the rate of adaptation of a particular process machine over time shows such a pattern. We propose that the type of micro-level understanding of learning by doing we have pursued in this paper can contribute to a better understanding of learning curves for entire production processes, since these learning curves are the aggregate of more micro-level changes.

Our discovery that some of the problems in our sample were caused by changes to the use environment introduced *after* introduction of the machine has an additional interesting implication for the innovation process. Stable problems with stable causes can eventually be gotten right, although, as we have seen, probably not without learning by doing. Dealing with this type of problem will involve viewing initial implementation as an extension of the innovation process [14]. For example, one might shift from product and service development methods that assume that one can specify a user need and use environment accurately at the start of a project to methods

[4] Rosenberg distinguished larning by doing and learning by using are distinguished by the context in which learning occurs ("...gains that are internal to the production process (doing) and gains that are generated as a result of subsequent use of that product (using)" [20, p. 122]) rather than by attributes of the learning process itself. On the face of it, we see no reason why the learning process will differ between these two contexts, and so suspect that our findings will apply to both.

such as rapid prototyping [5] that incorporate trial and error in the use environment into the development process.

But problems caused by changes in the use environment after introduction of the machine, primarily due to user learning by doing, will presumably continue to arise. This suggests that one can never get it right, and that innovation may best be seen as a continuous process, with particular product embodiments simply being arbitrary points along the way.

References

[1] Paul S. Adler and Kim B. Clark, Behind the Learning Curve: A Sketch of the Learning Process, *Management Science* 37 (3) (1991).

[2] Christopher Alexander, *Notes on the Synthesis of Form* (Harvard University Press, Cambridge, MA, 1964).

[3] Thomas J. Allen, Studies of the Problem-Solving Process in Engineering Design, *IEEE Transactions on Engineering Management* EM-13 (2) (1966) 72–83.

[4] Kenneth J. Arrow, The Economic Implications of Learning by Doing, *Review of Economic Studies* 29 (1962) 155–173.

[5] Robert Axelrod, *The Evolution of Cooperation* (Basic Books, New York, NY, 1984).

[6] Jonathan Baron, *Thinking and Deciding* (Cambridge University Press, New York, NY, 1988).

[7] Barry W. Boehm, Terence E. Gray, and Thomas Seewaldt, Prototyping Versus Specifying: A Multiproject Experiment, *IEEE Transactions on Software Engineering* SE-10 (3) (1984) 290–303.

[8] Barney G. Glaser and Anselm L. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research* (Aldine de Gruyter, Hawthorne NY,).

[9] Hassan Gomaa, The Impact of Rapid Prototyping on Specifying User Requirements, *ACM Sigsoft Software Engineering Notes* 8 (2) (1983) 17–28.

[5] Initially developed for use in software development, rapid prototyping is explicitly designed to shuttle repeatedly between manufacturer and user in order to better determine the 'real need' for a given software package. First, key functions of proposed software products are simulated (prototyped) and provided to users for trial. Users then experiment with these prototypes, and ask manufacturers for improvements based upon what they have learned. This back and forth process continues until users are satisfied. This approach has been found to be better at creating a good fit between need and solution than methods that rely upon the accuracy of an initial statement of need by users [7,9].

[10] K.F. Habermeier, Product Use and Product Improvement, *Research Policy* 19 (1990) 271–283.

[11] Robert L. Hayes and Kim B. Clark, Exploring the Sources of Productivity Differences at the Factory Level, in: K.B. Clark, R.L. Hayes and C. Lorenz (Editors), *The Uneasy Alliance* (HBS Press, Boston, MA, 1985) pp. 425–458.

[12] Ernest J. Henley and Hiromitsu Kumamoto, *Reliability Engineering and Risk Assessment* (Prentice Hall, Englewood Cliffs, NJ, 1981), chs 2, 3 and 7.

[13] Charles A. Lave and James G. March, *An Introduction to Models in the Social Sciences* (Harper and Row, New York, NY, 1975).

[14] Dorothy Leonard-Barton, Implementation as Mutual Adaptation of Technology and Organization, *Research Policy* 17 (1988) 251–265.

[15] David L. Marples, The Decisions of Engineering Design, *IRE Transactions on Engineering Management* (1961) 55–71.

[16] Kazuhiro Mishina, Learning by New Experiences, division of research working paper 92-084, Harvard Business School, Boston, MA, May 1992.

[17] Nuclear Regulatory Commission Report # 75/014, October 1975.

[18] Harry E. Pople, Jr. Heuristic Methods for Imposing Structure on Ill-Structured Problems: The Structuring of Medical Diagnostics, in: Peter Szolovits (Editor), *Artificial Intelligence in Medicine* (Westview Press, Boulder, CO, 1982) ch. 5.

[19] W.R. Reitman, *Cognition and Thought* (Wiley, New York, 1965).

[20] Nathan Rosenberg, *Inside the Black Box: Technology and Economics* (Cambridge University Press, New York, NY, 1982).

[21] H.A. Simon, The Structure of Ill Structured Problems, *Artificial Intelligence* 4 (1973) 181–201.

[22] Herbert A. Simon, *The Sciences of the Artificial*, 2nd edn (MIT Press, Cambridge, MA, 1981).

[23] Robert P. Smith and Steven D. Eppinger Identifying Controlling Features of Engineering Design Iteration, Sloan School of Management Working Paper # 3348-91-MS, December 1991.

[24] Marcie Tyre and Oscar Hauptman, Effectiveness of Organizational Response Mechanisms to Technological Change in the Production Process, *Organization Science* 3 (1992) 301–321.

[25] Marcie J. Tyre and Wanda Orlikowski, Windows of Opportunity: Temporal Patterns of Technological Adaptation, *Organization Science*, forthcoming.

[26] Marcie J. Tyre and Eric von Hippel, Locating Adaptive Learning: The Situated Nature of Adaptive Learning in Organizations, Sloan School of Management working paper #BPS 3568–93, May 1993.

[27] E. von Hippel, *The Sources of Innovation* (Oxford University Press, New York; 1988).

[28] T.P. Wright, Factors Affecting the Cost of Airplanes, *Journal of Aeronautical Science* 3 (February (1936) 122–128.