Parallel and Sequential Testing of Design Alternatives

Christoph H. Loch • Christian Terwiesch • Stefan Thomke
INSEAD, Boulevard de Constance, 77305 Fontainebleau Cedex, France
The Wharton School, University of Pennsylvania, Philadelphia, Pennsylvania 19104
Harvard Business School, Soldiers Field, Boston, Massachusetts 02163
christoph.loch@insead.fr • terwiesch@wharton.upenn.edu • sthomke@hbs.edu

An important managerial problem in product design in the extent to which testing activities are carried out in parallel or in series. Parallel testing has the advantage of proceeding more rapidly than serial testing but does not take advantage of the potential for learning between tests, thus resulting in a larger number of tests. We model this trade-off in the form of a dynamic program and derive the optimal testing strategy (or mix of parallel and serial testing) that minimizes both the total cost and time of testing. We derive the optimal testing strategy as a function of testing cost, prior knowledge, and testing lead time. Using information theory to measure the test efficiency, we further show that in the case of imperfect testing (due to noise or simulated test conditions), the attractiveness of parallel strategies decreases. Finally, we analyze the relationship between testing strategies and the structure of design hierarchy. We show that a key benefit of modular product architecture lies in the reduction of testing cost.
(*Testing; Prototyping; Learning; Optimal Search; Modularity*)

1. Introduction

Beginning with Simon (1969), a number of innovation researchers have studied the role of testing and experimentation in the research and development process (Simon 1969, Allen 1977, Wheelwright and Clark 1992, Thomke 1998, Iansiti 2000). More specifically, Simon first proposed that one could "think of the design process as involving, first, the generation of alternatives and, then, the testing of these alternatives against a whole array of requirements and constraints. There need not be merely a single generate-test cycle, but there can be a whole nested series of such cycles" (Simon 1969, p. 149).

The notion of "design-test" cycles was later expanded by Clark and Fujimoto (1989) to "design-build-test" to emphasize the role of building prototypes in design, and to "design-build-run-analyze" by Thomke (1998), who identified the analysis of a test or an experiment to be an important part of the learning process in product design. These results echoed earlier empirical findings by Allen (1977, p. 60) who observed that research and development teams he studied spent, on average, 77.3% of their time on experimentation and analysis activities that were an important source of technical information for design engineers. Similarly, Cusumano and Selby (1995) later observed that Microsoft's software testers accounted for 45% of its total development staff. Because testing is so central to product design, a growing number of researchers have started to study testing strategies, or, to use Simon's words once more, optimal structures for nesting a long series of design-test cycles (Cusumano and Selby 1995, Thomke and Bell 2001). Testing and iteration have been identified as variables reducing time-to-market, especially in industries of high uncertainty and rapid change. The accelerating effect of testing has been reported by Eisenhardt and

0025-1909/01/4705/0663$5.00
1526-5501 electronic ISSN

Tabrizi (1995) in the computer industry, as well as by Terwiesch and Loch (1999) across various sectors of the electronics industries.

Integral to the structure of testing is the extent to which testing activities in design are carried out in parallel or in series. Parallel testing has the advantage of proceeding more rapidly than serial testing, but does not take advantage of the potential for learning between tests—resulting in a larger number of tests to be carried out. As real-world testing strategies are combinations of serial and parallel strategies, managers and designers thus face difficult choices in formulating an optimal policy for their firms. This is particularly important in a business context where new and rapidly advancing technologies are changing the economics of testing.

The purpose of this paper is to study the fundamental drivers of parallel and sequential testing strategies and develop optimal policies for research and development managers. We achieve this by formulating a model of testing that accounts for testing cost and lead time, prior knowledge, and learning between tests. We show formally under which conditions it is optimal to follow a more parallel or a more sequential approach. Moreover, using a hierarchical representation of design, we also show that there is a direct link between the optimal structure of testing activities and the structure of the underlying design itself, a relationship that was first explored by Alexander (1964) and later reinforced by Simon (1969).

Our analysis yields three important insights. *First*, the optimal mix of parallel and sequential testing depends on the *ratio* of the (financial) cost and (cost of) time of testing: More expensive tests make sequential testing more economical. In contrast, slower tests make parallel testing more attractive for development managers (see §3).

Second, imperfect tests reduce the potential uncertainty reduction when testing design alternatives. Using information theory to measure test efficiency, we show that such imperfect tests decrease the attractiveness of parallel testing strategies (see §4).

Third, the structure of design hierarchy influences to what extent tests should be carried out in parallel or sequentially. We show that a modular product architecture can radically reduce testing cost compared to an integral architecture. We thus suggest a link between the extensive literature on design architecture and the more recent literature on testing (§5).

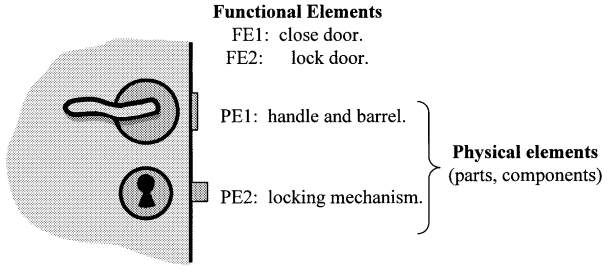
2. Parallel and Sequential Testing in Product Design

Design can be viewed as the creation of synthesized solutions in the form of products, processes, or systems that satisfy perceived needs through the mapping between functional elements (FEs) and physical elements (PEs) of a product. Functional elements are the individual operations and transformations that contribute to the overall performance of the product. Physical elements are the parts, components, and subassemblies that implement the product's functions (Ulrich and Eppinger 2000, see also Suh 1990, p. 27).

Assume that we are interested in designing the opening and closing mechanism of a door. To illustrate this view of product design, we consider only two of many possible FEs: the ability to *close* it (block it from randomly swinging open), with the possibility of opening from either side, and the ability to *lock* it (completely disallowing opening from one side or from both sides). The physical elements, or design alternatives, include various options of shape and material for the handle, the various barrels, and the lock (see Figure 1).

An integral characteristic of designing products with even moderate complexity is its *iterative nature*. As designers are engaged in problem solving, they iteratively resolve *uncertainty* about which physical elements satisfy the perceived functional elements. We will refer to the resolution of this uncertainty as a test or a series of tests.

Figure 1 FEs and PEs in the Design of a Door



It is well known that product developers generally do not expect to solve a design problem via a single iteration, and they often plan a series of design-test cycles or experiments, to bring them to a satisfactory solution in an efficient manner (Allen 1966, Simon 1969, Smith and Eppinger 1997, Thomke 1998). When the identification of a solution to a design problem involves more than one such iteration, the information gained from a previous test(s) may serve as an important input to the design of the next one. Design-test cycles that do incorporate learning derived from other cycles in a set are considered to have been conducted in series. Design-test cycles that are conducted according to an established plan that is not modified as a result of the finding from other experiments are considered to have been conducted in parallel.

For example, one might carry out a preplanned "array" of design experiments, analyze the results of the entire array, and then carry out one or more additional verification experiments as is the case in the field of formal "design of experiments (DOE)" methods (Montgomery 1991). The design-test cycles in the initial array are viewed as being carried out in parallel, while those in the second round are carried out in series with respect to that initial array. Such parallel strategies in R&D have been first suggested by researchers as far back as Nelson (1961) and Abernathy and Rosenbloom (1968), and more recently by Thomke et al. (1998) and Dahan (1998).

Specifically, there are three important factors that influence optimal testing strategies: cost, learning between tests, and feedback time. First, a test's *cost* typically involves the cost of using equipment, material, facilities, and engineering resources. This cost can be very high, such as when a prototype of a new car is used in destructive crash testing, or it can be as low as a few dollars, such as when a chemical compound is used in pharmaceutical drug development and is made with the aid of combinatorial chemistry methods and tested via high-throughput screening technologies (Thomke et al. 1998). The cost to build a test prototype depends highly on the available technology and the degree of accuracy, or fidelity, that the underlying model is intended to have (Bohn 1987). For example, building the physical prototype used in automotive crash tests can cost hundreds of thousands of dollars, whereas a lower-fidelity "virtual" prototype built inside a computer via mathematical modeling can be relatively inexpensive after the initial fixed investment in model building has been made.

Second, the amount of *learning* that can be incorporated in subsequent tests is a function of several variables, including prior knowledge of the designer, the level of instrumentation and skill used to analyze test results, and, to a very significant extent, the topography of the "solution landscape" that the designer plans to explore when seeking a solution to her problem (Alchian 1950, Kauffman and Levin 1987, Baldwin and Clark 1997a). In the absence of learning, there is no advantage in carrying out tests sequentially, other than meeting specific constraints that a firm may have (e.g., limited testing resources).

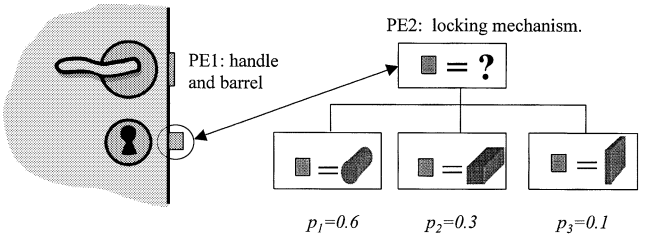
Third, the amount of learning is also a function of how timely feedback is received by the designer. It is well known that misperceptions and delays in feedback from actions in complex environments can lead to suboptimal behavior and diminished learning (Huberman and Hogg 1988, Bohn 1987). The same is true for noise that has been shown to reduce the ability to improve operations (Bohn 1995). Thus, the *time* it takes to carry out a test and obtain results not only allows design work to proceed sooner, but also influences the amount of learning between sequential tests.

3. A Model of Perfect Testing

We start our analysis by focussing on the optimal testing strategy in the design of one single physical element (PE). Consider, for example, the PE "locking mechanism" from Figure 1, for which there exist a number of design alternatives, depicted in Figure 2. Three different geometries of the locking barrel might fulfill the functional element (FE) "lock the door." Based on her education and her previous work, the design engineer forms prior beliefs, e.g., "a cylinder is likely to be the best solution; however, we might also look at a rectangular prism as an alternative geometry."

More formally, the engineer's prior beliefs can be represented as a set of probabilities p_i defined over

Figure 2 **Solutions for the PE "Locking Mechanism" to Fulfill the FE "Lock the Door"**



the alternatives $1 \cdots N$ where $p_i = \Pr\{\text{Candidate } i \text{ is the most preferred solution}\}$. To resolve the residual uncertainty, one geometry i is tested. Once the engineer can observe the result of the test, she gains additional information on whether or not this geometry is likely to be the most preferred solution available. If a test resolves the uncertainty corresponding to a solution candidate completely, we refer to this test as a *perfect test* (imperfect testing will be analyzed in §4). Based on a test outcome, the designer can update her beliefs. If the tested candidate turns out to be the most preferred, its probability gets updated to 1 and the other probabilities are renormalized accordingly. Otherwise, p_i is updated to 0. This updating mechanism represents learning in the model. It implies that a test reveals information on a solution candidate *relative* to the other candidates.

Of course, perfect testing applies only to problems where the test outcome can be defined in binary terms. A good example of such development problems is geometric fit. When different parts and/or subsystems occupy the same coordinates in three-dimensional space, they interfere with one another. These so-called interference problems are very common during the geometric integration of a complex product. As Boeing has learned over many decades, airplane development can involve hundreds of thousands of potential interferences that are identified through testing with computer-aided design models, prototypes, and during final assembly. Boeing managers define the degree to which such testing occurs in parallel or sequentially through the design of its development process. Each of those tests results in an "interference/no interference" result and, if the underlying model is accurate, allows the respective probabilities to be updated to 1 or 0. In such interference tests, there are two kinds of learnings that can be observed: (a) the actual outcome of the test (is there an interference?), and (b) information about other likely interferences that guides further downstream testing. In our paper, (a) relates to the efficiency of individual tests, whereas (b) relates to the learning mechanism between multiple rounds of tests.

Perfect testing also applies in contexts outside product development, for example, in the validation of a new piece of production equipment. Each of a number of potential root causes for a malfunctioning can be confirmed through a specific diagnosing test. A similar situation exists in medical diagnosis. Consider a patient who displays a certain symptom, e.g., hypotension. Each of a number of possible root causes can be confirmed through one specific and reliable diagnostic test, and multiple hypotheses may be tested sequentially or in parallel.

While the exact learning (updating) mechanism may differ from context to context, our model represents the real-world intermediate case between no learning (i.e., no useful information is revealed about which candidate should be tested in the next round) and perfect learning (i.e., a test reveals the direction towards the optimal solution, thus one test characterizes all candidates). As defined earlier, the presence of *some* learning between testing rounds is important in motivating the value of sequential tests. Thus, perfect learning would make parallel testing unnecessary, while a decrease in learning would clearly increase the attractiveness of parallel testing (we can show this as a special case of our model). Ignoring learning in our model would skew our results toward parallel search, and furthermore, learning has been identified as an important element of product development in the literature (Sobek et al. 1999). In our model, we hold learning between sequential rounds of testing constant, but vary a test's efficiency by including *imperfect* testing, i.e., the notation that a test does not fully reveal whether a design alternative is indeed the most preferred solution.

We assume that there is a fixed cost c per test, as well as a fixed lead time τ between the beginning of test-related activities and the observability of the

newly generated information. If lead time is important, as in the presence of a delay, it can be beneficial to order several tests in parallel. Let c_τ be the cost of delay for the time period of length τ. Testing thus "buys" information in the form of updated probabilities at the price of $nc + c_\tau$, where n is the number of tests the engineer orders in one period.

The problem of searching for a target in a search space with a probability distribution of the target's position in this space has long been studied in the fields of mathematics and computer science (e.g., Stone 1975). More recently, in the presence of computers equipped with multiple parallel processors, there has been a growing interest in the development of parallel search algorithms (Quinn 1987). Unlike our model, these algorithms take the number of parallel processors as exogenously given and constant over the computation time required for the problem. Our model, in contrast, considers the degree of parallelism as a decision variable that can dynamically be changed over the search.

Models of search and Bayesian learning have also been developed in statistics and economics literature under the label of sequential sampling procedures (sometimes also referred to as the "Secretary-," "Marriage-," or "Beauty-contest problem," see e.g., Degroot 1970). In these models, a decision maker needs to trade off a given cost of sampling a unit with the value of additional information. Given this one-dimensional cost-based (opposed to cost and time-based) approach to search, parallel search is not considered ($n = 1$). In a result known as "Pandora's rule," Weitzmann (1979) shows that if there are N "boxes" to be opened in a sequential search, box i offering a reward R with probability p_i, the box with the lowest "cost" $|A_i|c/p(A_i)$ should be opened first.[1] Here, $|A_i|$ is the number of objects in the box, c the search cost per object, and $p(A_i)$ the probability that the box contains the reward. Note that if all sets have equally many elements (in particular, if each solution candidate alone forms a set), this rule suggests to test the *most likely* candidate first.

[1] This review of Weitzman's result has been adapted to correspond to our situation. In our problem, we consider less general rewards than in Weitzman's Pandora's rule (in our model, a candidate is either right or wrong; there is no generally distributed reward).

However, Weitzman assumes that only one box can be opened at a time ($n = 1$), which ignores the aspect of testing lead time. In most testing situations, the designer not only needs to decide *which* test to run next, but also *how many* tests should be run in parallel. On the one hand, running many tests in parallel will result in diminishing returns. The value of running one additional test (uncertainty reduction) may be smaller than its incremental cost. For a development manager, this creates an interesting trade-off between cost and time, which we will now explore further.

The described testing problem can be seen as a dynamic program, where the state of the system is the set S of remaining potential solution candidates with their probabilities. The decision to be made in each stage of the dynamic program is the set of states to be tested next; call it A. The immediate cost of this decision is $|A|c + c_\tau$, and the resulting state is the empty set with probability $p(A) = \sum_{i \in A} p_i$, and it is $S - A$ with probability $\sum_{i \in (S-A)} p_i$. A testing policy is optimal for a given set of solution candidates with attached probabilities p_i, if it minimizes the expected cost (testing and delay) of reaching the target state $S = \{\}$.

THEOREM 1. *To obtain the optimal testing policy, order the solution candidates in decreasing order of probability such that $p_i \geq p_{i+1}$. Assign the first candidates to set A_1, the "batch" to be tested first, until its target probability specified in Equation* (2) *is reached. Assign the next candidates to set A_2 to be tested next (if the solution is not found in A_1), and so on, until all N leaves are assigned to n sets $A_1, \ldots, A_n$. The optimal number of sets*[2] *is*

$$n = \min\left\{N; \max\left\{1, \left[\frac{1}{2} + \sqrt{\frac{1}{4} + \frac{2cN}{c_\tau}}\right]\right\}\right\}, \qquad (1)$$

where $[\cdots]$ denotes the integer part of a number. The sets are characterized by their probabilities $p(A_i) = \sum_{j \in A_i} p_j$:

$$p(A_i) = \frac{1}{n} + \frac{c_\tau}{cN}\left(\frac{n+1}{2} - i\right) = \frac{2(n-i)}{n(n-1)}. \qquad (2)$$

It is interesting to note that the batch probabilities are described as a deviation from the average $1/n$: The first batches have a higher probability, the last batches

[2] The number of batches includes the last (nth) set, which is empty. Thus, the de facto number of sets is $n - 1$.

a lower probability than the average. Note that this does *not* imply that the number of solution candidates in the first batches tested is also higher: If probabilities initially fall off steeply with i, the first batch tested may have a lower number of solution candidates than the second batch. If the total number of candidates N is very large, the difference in probability among the batches shrinks.

The policy in Theorem 1 behaves as we would intuitively expect. When the testing cost c is very large, the batches shrink to 1, $n = N$, and testing becomes purely sequential to minimize the probability that a given candidate must be tested. If c_τ approaches infinity, n approaches 1: Testing becomes purely parallel to minimize time delay. When the total number of solution candidates N grows, the number of batches grows with $\sqrt{N}$. We describe this extreme behavior more precisely in the following corollary.

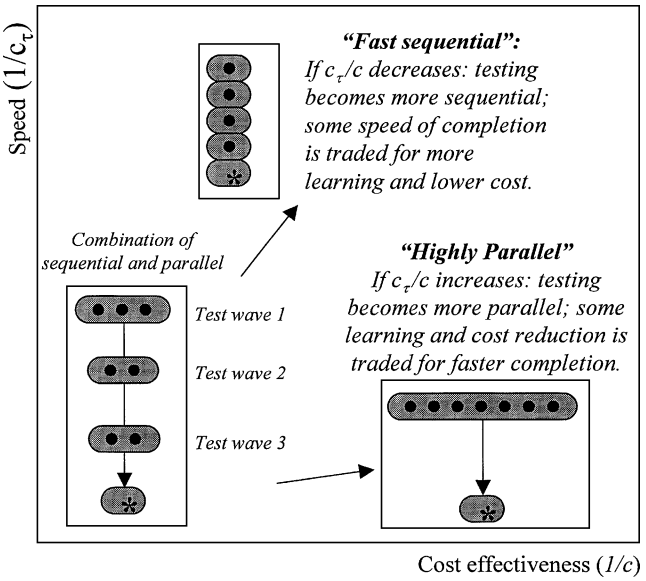
COROLLARY 1. *If* $1/N < c/c_\tau < (N+1)/2$, *the optimal expected testing time is* $(n+1)/3$, *and the expected total testing cost is* $(c_\tau(n+1)(3n+2))/12$. *If* $c/c_\tau \le 1/N$, *optimal testing is fully parallel* $(n = 1)$, *the testing time is* 1, *and the optimal total testing cost is* $(c_\tau + Nc)$. *If* $c/c_\tau > (N+1)/2$, *optimal testing is fully sequential, and the optimal total cost is* $\sum_i ip_i(c+c_\tau)$. *If all candidates are equally likely, this becomes* $((N+1)/2)(c+c_\tau)$.

In addition to defining the optimal testing policy, Theorem 1 provides an interesting structural insight concerning when to perform parallel search. Earlier studies have proposed that new testing technologies have significantly reduced the cost of testing, thus increasing the attractiveness of parallel strategies (e.g., Sobek et al. 1999, Terwiesch et al. 1999, Thomke 1998). Our results clearly demonstrate this—as test cost decreases, the optimal batch size goes up. For the extreme case of $c = 0$, the above corollary prescribes a fully parallel search. This is precisely what happened in the pharmaceutical industry when new technologies such as combinatorial chemistry and high-throughput screening reduced the cost of making and testing a chemical compound by orders of magnitude. Instead of synthesizing and evaluating, say, 5–10 chemical compounds per testing iteration, pharmaceutical firms now test for hundreds or thousands of compounds per test batch in the discovery and optimization of new drug molecules.

However, as the model shows, looking primarily at the cost benefits of new technologies ignores a second improvement opportunity. To fully understand the impact of new testing technologies on testing cost and search policy, one must consider that the results not only come at less cost, but that they also come in less *time*. In the automotive industry, for example, new prototyping technologies such as computer simulation or stereolithography have reduced the lead time of a test from months or weeks to days or hours. Thus, not only c changes, but also c_τ.

If both parameters change *simultaneously*, the amount of parallel testing might go down or up. This interplay between testing cost and information turnaround times is illustrated in Figure 3. The coordinates are speed $(1/c_\tau)$ and cost effectiveness $(1/c)$ of tests. The diagram in the lower left corner of the figure represents testing economics with relatively low speed and cost effectiveness, resulting in some optimal combination of parallel and sequential testing as described in Theorem 1. Moving toward the lower right of the figure corresponds to a reduction in testing cost, moving up to a reduction in testing time (or urgency). If a testing cost improvement outweighs a time improvement, the test batches should grow and search becomes more parallel, as in the pharmaceutical example above.

Figure 3 Impact of Test Speed and Cost on Testing Strategy



If, in contrast, the dominant improvement is in the time dimension, the faster feedback time allows for learning between tests. The optimal search policy becomes "fast-sequential." In this case, total testing cost *and* total testing time can decrease: total testing time because of shorter test lead times and total testing cost because of "smarter" testing (based on the learning between tests, resulting in fewer wasted prototypes). Thus, in the evaluation of changing testing economics, a purely cost-based view may lead to an erroneous conclusion.

4. Imperfect Testing

Real-world testing is often carried out using simplified models of the test object (e.g., early prototypes) and the expected environment in which it will be used (e.g., laboratory environments). This results in imperfect tests. For example, aircraft designers often carry out tests on possible aircraft design alternatives using scale prototypes in a wind tunnel—an apparatus with high wind velocities that partially simulate the aircraft's intended operating environment. The value of using incomplete prototypes in testing is two-fold: to reduce investments in aspects of "reality" that are irrelevant for the test, and to control out noise to simplify the analysis of test results. We model the effect of incomplete tests and/or noise as *residual uncertainty* that remains after a design alternative has been tested (Thomke and Bell 1999). Such a test will be labeled as *imperfect*.

Only one candidate can be the most preferred ($i = 1$); all others must be less preferred ($k = 0$ for all k not equal to i). The tester does not know initially which candidate is preferred. We assume that a test of design candidate i gives one of only two possible signals: $x = 1$ indicates "candidate i is the most preferred design," and $x = 0$ indicates "candidate i is not the most preferred design." An imperfect test is characterized by its error probabilities: The false negative occurs with $\Pr\{x = 0 \mid i = 1\} = 0.5(1 - \gamma)$, and the false positive with $\Pr\{x = 1 \mid i = 0\} = 0.5(1 - \beta)$. The test fidelity γ captures the power of the test in identifying a winning design candidate when it is tested ($\gamma \in [0, 1]$, from uninformative to fully informative). Similarly, the fidelity β represents the power of the test in correctly eliminating an inferior candidate when it is tested. This implies the following marginal probabilities of the signal from testing candidate i with fidelities γ and β:

$$\begin{aligned} \Pr\{x_i = 1\} &= \frac{1}{2}[1 - \beta + (\gamma + \beta)p_i]; \\ \Pr\{x_i = 0\} &= \frac{1}{2}[1 + \beta - (\gamma + \beta)p_i]. \end{aligned} \tag{3}$$

The posterior probabilities of all design candidates can be written as (j not tested):

$$\begin{aligned} p_i(x_i = 1) &= \frac{(1 + \gamma)p_i}{1 - \beta + (\gamma + \beta)p_i}; \\ p_i(x_i = 0) &= \frac{(1 - \gamma)p_i}{1 + \beta - (\gamma + \beta)p_i}; \end{aligned} \tag{4}$$

$$\begin{aligned} p_j(x_i = 1) &= \frac{(1 - \beta)p_j}{1 - \beta + (\gamma + \beta)p_i}; \\ p_j(x_i = 0) &= \frac{(1 + \beta)p_j}{1 + \beta - (\gamma + \beta)p_i} \quad (j \neq i). \end{aligned} \tag{5}$$

The fact that *all* probabilities are updated after testing candidate i represents learning. If a test is perfect ($\gamma = \beta = 1$), these posterior probabilities describe the perfect testing in the previous subsection. If a test is not perfect, it only *reduces the uncertainty* about a design alternative. It takes an infinite number of tests to reduce the uncertainty to zero (bring one p_k to 1). Therefore, the designer can only strive to reduce uncertainty of the design to a "sufficient confidence level $(1 - \alpha)$" in the design, where one $p_k \geq (1 - \alpha)$, and $\sum_{j \neq k} p_j \leq \alpha$. This is one of the reasons why a designer "satisfices," as opposed to optimizes, a product design (Simon 1969).

We first concentrate on a situation where only one alternative can be tested at once (sequential testing, Theorem 2a), turning to testing several alternatives in parallel afterward (Theorem 2b). The designer's problem is to find a testing sequence that reaches a sufficient confidence level at the minimum cost. As all information available to the designer is encapsulated in the system state $S = \mathbf{p} = \{p_1, \ldots, p_N\}$ and the transition probabilities (4) and (5) depend only on S,

we can formulate the problem as a dynamic program: At each test, pay an immediate cost of $(c+c_\tau)$ (for executing the test and for the time delay). Find a policy $\pi(\mathbf{p})$ that chooses a solution candidate $i \in \{1, \ldots, N\}$ to minimize:

$$V(\mathbf{p}) = (c+c_\tau) + \text{Min}_i\{\Pr\{x_i=1\}V(p_i(x_i=1); \; p_j(x_i=1)\ \forall\ j\neq i) + \Pr\{x_i=0\}V(p_i(x_i=0); \; p_j(x_i=0)\ \forall\ j\neq i)\}, \quad (6)$$

where $V(\mathbf{p})=0$ if and only if a design of sufficient confidence level has been found. While we cannot write down the optimal testing cost for this problem, we can identify the optimal policy. In many, but not all, cases it has the same structure as for perfect tests.

THEOREM 2A. *If testing is performed sequentially, that is, one design alternative at a time, it is optimal to always test the candidate with p_i closest to*:

$$p^* = \left[\beta - \frac{2^{[f(\gamma)-f(\beta)]/(\gamma+\beta)}-1}{2^{[f(\gamma)-f(\beta)]/(\gamma+\beta)}+1}\right] \Big/ (\gamma+\beta). \quad (7)$$

This is equivalent to testing the most likely candidate (with the largest p_i) whenever $\gamma \geq \beta$. If the false negative fidelity $\beta > \gamma$, it may be optimal to test the second-most likely candidate. ($f(\cdot)$ is characterized in the proof.)

Standard dynamic programming techniques cannot establish optimality of a myopic policy as stated in the theorem because the transition probabilities are state dependent. Therefore, we use information theory as a tool to express the uncertainty reduction offered by imperfect tests (Suh 1990, Reinertsen 1997). This theory is based on Shannon (1948) and states that the *entropy* of a system indicates the amount of "choice" or uncertainty in that system. In particular, we define the *entropy* of the ith design alternative and the entropy of the entire design problem, respectively, as

$$H_i = -p_i \log p_i, \quad H = \sum_i H_i. \quad (8)$$

The entropy captures knowledge about the alternative intuitively: It is maximal when $p_i = 1/2$, in which case $H_i = \log 2 = 1$ bit. That is, the uncertainty about design alternatives is maximal when all alternatives are equally likely to be the solution. $H_i = 0$ if $p_i = 0$ or if $p_i = 1$, that is, if it is known precisely whether the candidate leads to the solution or not. The entropy H of the entire problem measures the uncertainty of the entire design. It is jointly concave in the p_i and maximal at $N \log N = N$ bits if all candidates are equally likely to be the best solution. $H = 0$ if and only if there is one candidate k with $p_k = 1$ (and thus, all other candidates are eliminated). Using the design problem's entropy, we can prove the theorem (see Appendix).

Theorem 2a shows in what way imperfect sequential testing is more complex than perfect testing. First, the policy is dynamic—while the assignment of testing candidates to time periods can be done ex ante for perfect testing, an initially likely candidate may become unlikely (or vice versa) if probabilities are updated imperfectly. This is why Theorem 2a provides a dynamic policy. Second, testing the most likely candidate in each round remains true in general only if the fidelity of identifying a winning candidate (γ) is at least as high as the fidelity of correct elimination (β). If $\beta \gg \gamma$, it is less error prone to eliminate a candidate than to declare a winner. In this case, it may be better to test the second-most likely candidate if its probability is closer to 45%, while the most likely candidate has a probability of close to 55%. However, this situation applies only in a small part of the (γ, β) space, and a numerical evaluation of (7) shows that even here, the efficiency loss from testing the most likely candidate is small (details are shown in the proof). Thus, we can conclude that the policy of always testing the currently most likely candidate is robust in practice.

We now relax the condition of sequentiality and allow the simultaneous testing of several design alternatives. We exclude multiple simultaneous tests of the *same* alternative.[3] We assume that the outcome of testing alternative i depends only on its own properties, but not on any other alternative. The test outcomes are independent because of simultaneity—no learning takes place until after a test iteration has been completed. For parallel imperfect testing of this kind, we can prove the following result.

[3] The situation does not correspond to, for example, consumer focus groups, where the same design alternative is shown to different consumers (which would increase the fidelity of the test).

THEOREM 2B. *Assume n different design alternatives are tested simultaneously as described above. It is optimal to always test the alternatives with the largest probabilities p_i. A higher number of parallel tests, n, reduces the entropy with diminishing returns, and there is an optimal number of parallel tests. The optimal number of parallel tests increases when either of the fidelities γ or β increases.*

Theorem 2b shows that when multiple candidates are tested in parallel, the policy of always choosing the most likely ones is robust (for one test, we saw that the second-most likely might be chosen, but both most likely ones are always included in parallel testing). In addition to the cost ration c/c_τ from Theorem 1, Theorem 2b identifies another reason why parallel testing may be more economical. A higher testing fidelity enhances the uncertainty reduction that can be gained from multiple tests. Therefore, the number of parallel tests that justify their investment c increases.

5. Testing and the Structure of Design Hierarchy

A number of researchers have studied the role of design structure in the innovation process and have found it to matter significantly (Baldwin and Clark 1997a, Clark 1985, Marples 1961, Smith and Eppinger 1997, Simon 1969, Ulrich 1995). More specifically, it has been proposed that designs with smaller subsystems that can be designed and changed independently but function together as whole—a structure often referred to as modular—can have far-reaching implications for firm performance, including the management of product development activities. This approach was first explored by Alexander (1964) and was later reinforced by Simon (1969, 1981): "To design [such] a complex structure, one powerful technique is to discover viable ways of decomposing it into semi-independent components corresponding to its many function parts. The design of each component can then be carried out with some degree of independence of the design of others, since each will affect the others largely through its function and independently of the details of the mechanisms that accomplish the function" (Simon 1969, p. 148). In this section, we will explore the relationship between design structure and optimal testing.

A simple search model might capture the testing process related to one single physical element (PE) and a single functional element (FE), but in general, product design is concerned with more complex systems. The *design structure* links the product's various FEs to its PEs. In the case of an uncoupled design, each FE is addressed by exactly one PE. In coupled designs, the mapping from FEs to PEs is more complex.

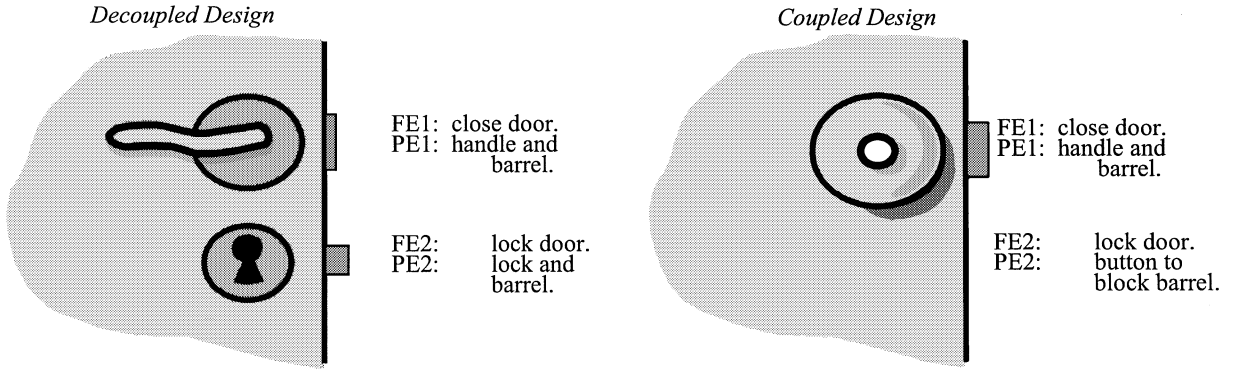
Consider the two different door designs illustrated in Figure 4. The design on the left of Figure 4 is *uncoupled*, that is, each FE is addressed by exactly one physically separate component. Closing is performed by a handle that moves a blocking barrel (which inserts into the door frame), and locking is carried out by turning a key that moves a second barrel. If the design is uncoupled, each FE is fulfilled by one PE, and each PE contributes to one FE. We call this separation of FEs *functional independence* of the design.[4] Designs that are functionally independent are also called *modular* (Ulrich and Eppinger 2000).

The design on the right in Figure 4 is *coupled*. Closing is implemented by a doorknob, the turning of which moves a blocking barrel. Locking is enacted by a button in the center of the doorknob that blocks the doorknob from turning. The locking function uses *both* physical components, in particular, the same rod moving the barrel when opening/closing the door is blocked from moving when locking the door.

The architecture of the product has a fundamental influence on the testing process. In the case of functional independence between closing and locking the door, the corresponding subsystems (PEs) can be tested independently. If there are three candidates for the barrel (Figure 4) and two candidates for the lock, a total of $3+2=5$ tests would cover the total search space. If, however, the closing and locking are coupled, testing requires a specification of both PEs, closing barrel *and* locking barrel. If the outcome of the test

[4] In addition to functional dependencies, elements can also dependent on each other because of their physical attributes which we will refer to as technical dependence. The interdependence between PEs can be captured in the *design structure matrix* (DSM) (Steward 1981, Eppinger et al. 1994).

Figure 4 Functional Decoupling in the Design of a Door



is negative (FEs were not fulfilled), learning from the failure is more complex. For example, if the closing FE was fulfilled, but not the locking FE, the engineer cannot infer whether she should just change the locking barrel, or also the closing barrel. An exhaustive search requires $3^*2 = 6$ tests.[5]

An intermediate case between coupled design and uncoupled design results, if the PEs contributing to the first FE can be determined without specifying the PEs contributing to the second FE, but not vice versa. In this case, we speak of *sequential dependence*, and it is possible to test the first PE/FE before addressing the second.

We see that functional and technical structure influences testing in two ways. First, it influences the number of tests required for an exhaustive search (3^*2 vs. $3+2$ in the door example). Second, it influences the timing of the tests. If the design is uncoupled, tests can be done in parallel (without any additional cost). In the case of sequential dependence, parallel testing is possible, but only up to a certain level. Coupled designs, however, cause the search space to grow exponentially without opportunities for parallel testing (other than the parallel testing where the designer precommits to several prototypes at once). The resulting effect of product architecture on testing cost is analyzed in Theorem 3 below.

For simplicity of exposition, assume a symmetric situation where each PE has N solution alternatives of equal probability, and there is one PE for each of M functional requirements. We consider the three generic architectures independent (modular), sequentially dependent (any two PEs have an upstream-downstream relationship), or integrated (each PE impacts all other PEs). Clearly, most complex systems include aspects of all three of these categories, but in the interest of a clear comparison, it is most useful to analyze them as three distinct types along a spectrum of structural possibilities.

THEOREM 3. *Suppose a design has M PEs with N equally likely solution candidates each, and a test costs c and takes one time unit costing c_τ. Then the expected testing costs for the three architectures are (where $n(N) = 1/2+\sqrt{1/4+2cN/c_\tau}$ from Theorem 1):*

	Parallel, $n(N)=1$ $\left(\frac{c}{c_\tau} \le \frac{1}{N}\right)$	Intermediate $\left(\frac{1}{N} < \frac{c}{c_\tau} < \frac{N+1}{2}\right)$	Sequential, $n(N)=N$ $\left(\frac{N+1}{2} \le v\frac{c}{c_\tau}\right)$
$\mathbf{C}_{mod} =$	$c_\tau + NMc$	$\le Mc_\tau\left[\frac{n(N)}{M+1} + \frac{(n(N)+1)(n(N)-2/3)}{12}\right]$	$= \frac{MN}{M+1}(c_\tau + c)$
$\mathbf{C}_{sequ} =$	$Mc_\tau + NMc$	$Mc_\tau \frac{(n(N)+1)(n(N)-2/3)}{12}$	$\frac{N+1}{2}M(c_\tau + c)$
$\mathbf{C}_{int} =$	$c_\tau + N^M c$ $\left(\text{if } \frac{c}{c_\tau} \le \frac{1}{N^M}\right)$	$\frac{c_\tau}{12}\left[n(N^M)+1\right]\left[n(N^M)-\frac{2}{3}\right]$ $\left(\text{if } \frac{1}{N^M} < \frac{c}{c_\tau} < \frac{N^M+1}{2}\right)$	$\frac{N^M+1}{2}(c_\tau + c)$ $\left(\text{if } \frac{N^M+1}{2} \le \frac{c}{c_\tau}\right)$

[5] Simon (1969) illustrates this point very nicely with the following example, which was originally supplied by W. Ross Ashby. "Suppose that the task is to open a safe whose lock has 10 dials, each with 100 possible settings, numbered from 0 to 99. How long will it take to open the safe by a blind trial-and-error search for the correct setting? Since there are 100^{10} possible settings, we may expect to examine about one half of these, on average, before finding the correct one [...]. Suppose, however, that the safe is defective, so that a click can be heard when any one dial is turned to the correct setting. Now each dial can be adjusted independently and does not need to be touched again while the others are being set. The total number of settings that have to be tried is only 10×50, or 500."

COROLLARY 2. *The testing costs depend on the product architecture as follows*: $C_{mod} < C_{sequ} \leq C_{int}$.

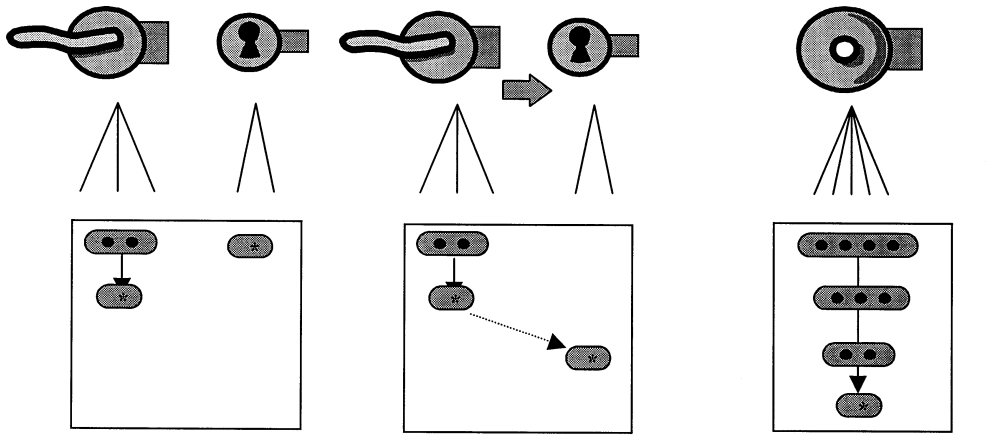
The theorem shows that in a modular architecture, the expected testing costs grow *sublinearly* with the number of PEs; the costs grow *linearly* in a sequentially dependent architecture, and they grow *exponentially* in an integrated architecture. Thus, the theorem explains that *testing effort* contributes to the benefits of a modular product architecture, simplifying the development process and often leading to lower total development time and cost. Figure 5 summarizes the connection between architecture and testing.

The results of Theorem 3 are consistent with similar propositions in the literature. Ulrich (1995) noted that for modular architectures, the design of each module can proceed almost independently and in parallel. System-level product testing would be limited to detecting unanticipated interactions, or areas where the system is not perfectly modular. The result of our analysis shows this to be true if modularity can be established in the functional and physical domains and if there is a direct one-to-one mapping between functional and physical elements (FEs and PEs). In such an extreme case of functional and technical modularity, there is no need for system-level testing. However, if there is at least one FE that is impacted by all PEs, the benefits of modularity are substantially reduced. In fact, all design alternatives and their impact on this FE would have to be considered for testing—a number that would increase very rapidly as Theorem 3 shows. If designers know little about functional (customer) elements and their interactions—not an unusual real-world dilemma—the value of modularity in testing quickly disappears. Indeed, as Baldwin and Clark (1997b) have shown, the presence of many modules can lead to a combinatorial explosion of testing and experimentation if the system-level impact on markets (or, in our definition, on the functional user domain) is highly uncertain.

6. Conclusion

In this paper, we have shown that the extent to which testing activities are carried out in parallel and series can have a significant impact on design performance. Parallel testing has the advantage of proceeding more rapidly than serial testing but does not take advantage of the potential for learning between tests, thus resulting in a larger number of tests. We model this trade-off in the form of a dynamic program and

Figure 5 Impact of Architecture on Testing



Independent
Low number of alternatives to be tested (2 + 3); PEs can be tested independently in parallel. Lowest testing cost and time.

Sequentially dependent
*Increase of tests from combination (to 2 * 3) can be avoided by testing downstream PE after upstream. Time increase avoids cost increase.*

Integral
*Increase of tests from combination to 2 * 3. Testing problem is most complex; highest testing cost and time.*

derive the optimal testing strategy (or mix of parallel and serial testing) that minimizes both, the total cost of testing and time. More specifically, our paper shows three results.

First, the optimal mix of parallel and sequential testing depends on the *ratio* of testing cost and time: More expensive tests make sequential testing more economical. In contrast, slower tests or an increasing opportunity cost of time make parallel testing more attractive for development managers.

Second, imperfect tests weaken the potential uncertainty reduction of each individual test. A lower test efficiency reduces the attractiveness of parallel testing. This is particularly important for managers who consider switching to early and less complete prototypes and/or the use of less controlled test environments.

Third, the design structure influences to what extent tests should be carried out in parallel or sequentially. We show that an important benefit of a modular product architecture comes from reduced testing cost, because it allows parallel testing without an increase in the number of test combinations. Thus, architecture can be an important lever for decreasing test cost.

As part of our testing model, we were also able to extend an important search model developed by Weitzman (1979). Whereas Weitzman studied sequential search, we included the option of carrying out search (or, in our case, testing) in parallel. We derived policies that would not only prescribe an optimal sequence of search but also inform decisionmakers about the degree to which such searches should be carried out in parallel. We expanded our analysis to include imperfect testing (less uncertainty reduction), using principles from information theory. Our last theorem also confirmed that there is an important relationship between two streams of research (design structure and testing) that we tried to establish more formally.

To conclude this paper, we propose three promising directions for further research that build directly on the findings presented here. First, it has been empirically observed that iterative testing can not only influence development cost and time, but also the quality of the design solution. It has been found that less costly and faster iterations through advanced technologies such as computer simulation can actually result in more experimentation, leading to novel solution concepts that could not be reasonably tested for with slower and more costly technologies. In the present paper, we have focused on the cost and time aspects, holding design solution quality constant, but it is possible to make N, the number of design alternatives tested, an explicit decision variable in our model. Expanding the search space will increase the testing costs, but also improve the design quality (possibly with diminishing returns). This future work relates our current model to the literature on set-based product development (e.g., Sobek et al. 1999).

Second, in the case of sequential dependence, it might be beneficial to start the testing of the second module before the testing for the first module has been finalized, i.e., to overlap the two testing processes in the spirit of concurrent engineering (Loch and Terwiesch 1998). Finding the optimal level of overlap between tests is thus a second opportunity for further research.

Third, we have shown learning between tests to be the primary advantage of sequential testing. In this paper, we have modeled the consequences of learning (uncertainty reduction through probability updating), but have not explicitly taken advantage of what is known about the different factors that influence learning. For example, a solution landscape represents the arena that the designers search to identify a solution to their problem. The probability of finding a solution increases as one ascends the "hills" in the landscape, and so the designer's goal is to devise a series of tests that will enable them to identify and explore those hills in an efficient manner. The amount that can be learned between tests then relates directly to the topography of the landscape. Very little can be learned by a designer about the direction of her search if, for example, the solution landscape is absolutely flat for all combinations except the correct one. In contrast, suppose that the solution landscape is a hill with only a single peak and sides that extend to all edges of the landscape.[6] In such a case a strategy of serial

[6] This is the shape, for example, of the value landscape in the children's game in which a child is guided to a particular spot via feedback from other children who say "warmer" each time a step is taken towards that spot.

testing may be the most efficient choice, because the information gained from each step taken is so useful in guiding the direction of the next trial step that the correct solution is often found after only a few trials.

Certainly, knowledge about the topology of solution landscapes will make sequential testing more attractive to designers. It is thus not surprising that well-studied engineering design problems tend to follow more sequential plans than, say, the early search for drug candidates in a relatively unknown solution space such as Alzheimer's disease, even after the cost and time of each test is accounted for. Some of these factors that influence learning can be included more explicitly in our model of parallel and sequential testing and thus provide further leverage in the formulation of optimal testing strategies for superior product development performance.

Acknowledgments

We thank two anonymous referees for thorough and thoughtful comments, which have helped to substantially improve this article.

Appendix

Proof of Theorem 1. If the solution is not found within the set A_i, the next set must be tested, which happens with probability $(1-\sum_{j=1}^{i} p(A_j))/(1-\sum_{j=1}^{i-1} p(A_j))$ (the denominator updates the probabilities of the remaining sets to sum to 1). Thus, we can write the total expected search cost as

$$\begin{aligned} EC &= |A_1|c + c_\tau + (1-p(A_1)) \\ &\quad \times \left[|A_2|c + c_\tau + \frac{1-p(A_1)-p(A_2)}{1-p(A_1)} \right. \\ &\quad \left. \times \left[|A_3|c + c_\tau + \frac{1-p(A_1)-p(A_2)-p(A_3)}{1-p(A_1)-p(A_2)}[\cdots] \right] \right] \\ &= \sum_{i=1}^{n} \left[(|A_i|c + c_\tau) \sum_{j=i}^{n} p(A_j) \right]. \end{aligned} \tag{A1}$$

The fact that the design alternatives should be assigned in decreasing order of probability follows from an exchange argument: Assume that there are two alternatives $j \in A_i$ and $k \in A_{i+1}$ with $p_k > p_j$. Exchange the two (test k before j). The resulting change in total expected cost is, from (A1), $(c_\tau + |A_{i+1}|)(p_j - p_k) < 0$. Thus, the candidates should be assigned as stated.

To simplify exposition, assume from now on that N is sufficiently large and the p_i small enough to approximate them by a continuous distribution function F. Now we transform the space, considering instead of the set sizes $|A_i|$ their probabilities $a_i = F(A_i) - F(A_{i-1})$, with $\sum_i a_i = 1$. The set sizes a_i correspond to fractions of N. In the transformed space, the solution candidates have a uniform probability density of 1, and the testing cost becomes Nc because the number of candidates has been compressed from N to 1. We can now state the objective function to be minimized (where we do not need to treat n as a decision variable as it is determined by the a_i):

$$\text{Min}_{a_i} \quad \sum_{i=1}^{n} (c_\tau + a_i Nc) \left(1 - \sum_{j=1}^{i-1} a_j \right) \tag{A2}$$

$$\text{subject to} \quad \sum_j a_j = 1; \quad a_i \geq 0 \ \forall \ i. \tag{A3}$$

The Lagrangian of this objective function is $L = c_\tau \sum_i i a_i + Nc \sum_i a_i (1 - \sum_{j=1}^{i-1} a_j) - \lambda(1 - \sum_i a_i) - \sum_i \mu_i a_i$. The optimality conditions for the Lagrangian are $a_i(\partial L/\partial a_i) = 0 \ \forall \ i$, $\partial L/\partial \lambda = 0$, and $\mu_i(\partial L/\partial \mu_i) = 0 \ \forall \ i$. These, in turn, yield the condition

$$c_\tau k + Nc a_k + \lambda = 0 \quad \text{for all } k, \text{ such that } a_k > 0. \tag{A4}$$

Condition (A4), first, implies that the second-order condition is fulfilled (differentiating it with respect to a_k gives $Nc > 0$, so the solution found is a cost minimum). Second, (A4) implies that the sets a_k are decreasing in size over k, so the first n^* sets are nonempty, and then no more candidates are assigned. Adding Equation (A4) over all k and considering that $\sum_k a_k = 1$ allows determining λ, and substituting in λ yields the optimal set probability (2). Finally, when the set probabilities are known, we can use the fact that $a_{n^*} > 0$ and $a_{n^*+1} \leq 0$ to calculate the optimal number of sets described in Equation (1). If $n^* \geq N$, then every solution candidate is tested by itself, which yields the largest number of sets possible.

Proof of Theorem 2a. We prove the theorem in three steps.

Step 1. Proxy Problem of Entropy Reduction. One-step entropy minimization. As the immediate reward in the dynamic program $-(c + c_\tau)$ is constant, the problem is to minimize the expected number of steps to go from the initial state to $V = 0$ (Bertsekas 1995, p. 300). Consider the proxy problem of minimizing the number of steps to go from $H(\mathbf{p})$ (uniquely determined by $\mathbf{p}$) to some target entropy H_0. After testing design alternative i, we can write the posterior entropy as (where "$x_i = a$" is abbreviated as "a"):

$$\begin{aligned} H_{post} &= -\frac{1}{2} \left\{ (1+\gamma) p_i \log \left(\frac{(1+\gamma)p_i}{1-\beta+(\gamma+\beta)p_i} \right) \right. \\ &\quad + \sum_{j \neq i} (1-\beta) p_j \log \left(\frac{(1-\beta)p_j}{1-\beta+(\gamma+\beta)p_i} \right) \\ &\quad + (1-\gamma) p_i \log \left(\frac{(1-\gamma)p_i}{1+\beta-(\gamma+\beta)p_i} \right) \\ &\quad \left. + \sum_{j \neq i} (1-\beta) p_j \log \left(\frac{(1+\beta)p_j}{1+\beta-(\gamma+\beta)p_i} \right) \right\} \end{aligned} \tag{A5}$$

$$\begin{aligned} &= H(\mathbf{p}) + \frac{1}{2} \{ p_i f(\gamma) + (1-p_i) f(\beta) \\ &\qquad - f(|\beta - (\gamma+\beta)p_i|) \}, \end{aligned} \tag{A6}$$

where $f(\varepsilon) = -(1+\varepsilon)\log(1+\varepsilon) - (1-\varepsilon)\log(1-\varepsilon)$ for $\varepsilon \in [0, 1)$. It can easily be shown that $f'(\varepsilon) < 0$ and $f''(\varepsilon) < 0$. We can thus calculate the first and second derivatives of H_{post} as a function of p_i,

which shows that this function is convex. Thus, the FOC characterizes a minimum for the posterior entropy, yielding Equation (7). p^* in (7) is a function of γ and β only; it increases in γ and decreases in β. Whenever $\gamma = \beta$, $p^* = 1/2$; moreover, the range of p^* is between 0.411 when $(\gamma, \beta) = (0.01; 0.99)$ and 0.59 when $(\gamma, \beta) = (0.99; 0.01)$. The largest one-step entropy reduction is produced by choosing the p_i closest to p^*. When $p^* = 1/2$, this is equivalent to p_i being the largest: If all $p_j \le 1/2$, this is true trivially. If one $p_k > 1/2$, then $p_k - 1/2 = 1/2 - \sum_{j \ne k} p_j$ which implies that p_k is closer to $1/2$ than all the p_j. If $\beta \gg \gamma$, it is possible that the second-most-likely candidate should be tested: Suppose $p^* = 0.411$, $p_1 \ge 0.42$, and $p_2 = 0.41$. Then p_2 yields the larger entropy reduction. Note that the third-largest p_k must be smaller than 0.17; that is, the deviation from "test the most likely" is only relevant in special cases where two candidates dominate and are about equally likely.

Step 2. Optimal Stationary Policy in Proxy. To establish optimality, we examine the expected *two-step* entropy reduction assuming that candidates i and then k are tested, while $j \ne i, k$ refers to all remaining candidates. Four cases result, of the test signals being $(1,1)$, $(1,0)$, $(0,1)$, and $(0,0)$. Because of renormalization, the updated probabilities are arithmetically messy, and we leave them to the reader (or they can be obtained from the authors). The resulting two-step posterior entropy H_{post^2} becomes:

$$
\begin{aligned}
-\frac{1}{4}\Bigg\{ & (1-\beta)(1+\gamma)p_i \log\left[\frac{(1-\beta)(1+\gamma)p_i}{A}\right] \\
& + (1-\beta)(1+\gamma)p_k \log\left[\frac{(1-\beta)(1+\gamma)p_k}{A}\right] + \sum_j (1-\beta)p_j \\
& \times \log\left[\frac{(1-\beta)^2 p_j}{A}\right] + (1+\beta)(1+\gamma)p_i \log\left[\frac{(1+\beta)(1+\gamma)p_i}{B}\right] \\
& + (1-\beta)(1-\gamma)p_k \log\left[\frac{(1-\beta)(1-\gamma)p_k}{B}\right] \\
& + \sum_j (1+\beta)(1-\beta)p_j \log\left[\frac{(1+\beta)(1-\beta)p_j}{B}\right] \\
& + (1-\beta)(1-\gamma)p_i \log\left[\frac{(1-\beta)(1-\gamma)p_i}{C}\right] \\
& + (1+\beta)(1+\gamma)p_k \log\left[\frac{(1+\beta)(1+\gamma)p_k}{C}\right] \\
& + \sum_j (1+\beta)(1-\beta)p_j \log\left[\frac{(1+\beta)(1-\beta)p_j}{C}\right] \\
& + (1+\beta)(1-\gamma)p_i \log\left[\frac{(1+\beta)(1-\gamma)p_i}{D}\right] \\
& + (1+\beta)(1-\gamma)p_k \log\left[\frac{(1+\beta)(1-\gamma)p_k}{D}\right] \\
& + \sum_j (1+\beta)^2 p_j \log\left[\frac{(1+\beta)^2 p_j}{A}\right]\Bigg\}, \qquad \text{(A7)}
\end{aligned}
$$

where $A = (1-\beta)(1-\beta+(\gamma+\beta)(p_i+p_k))$; $B = (1-\beta)(1+\beta) + (\gamma+\beta)[(1+\beta)p_i - (1-\beta)p_k]$; $C = B$ with i and k exchanged, and $D = (1+\beta)(1+\beta-(\gamma+\beta)(p_i+p_k))$. Inspection shows that H_{post^2} is the same when the order of testing i and k is exchanged. By induction, this implies that any order of testing a given collection of candidates gives in expectation the same posterior entropy. Step 1 and Step 2 together imply that it is optimal for the entropy proxy problem to test the p_i that is closest to p^* in all rounds.

Step 3. Optimality for the Dynamic Program. Set a target entropy that is more stringent than reaching $V = 0$, for example, $H_0 = -\alpha \log \alpha - (1-\alpha)\log(1-\alpha)$. Stop whenever one p_k reaches $(1-\alpha)$. As the policy derived in Steps 1 and 2 is optimal for *any* target entropy is also optimal when the entropy at this moment is used as a target. □

Proof of Theorem 2b. When we test design alternatives $i = 1, \ldots, n$ in parallel, our independence assumption implies that test outcome x_i is determined by (3), no matter what the other alternatives and tests are. Consider an arbitrary profile of test signals $(x_1, \ldots, x_n)$, where k is the number of tests that give a positive signal, $n-k$ is the number of tests that give a negative signal. Recall that it is impossible that more than one of the alternatives is in fact the right one. The marginal probability of the profile x is:

$$\Pr(\mathrm{x}) = \frac{(1-\beta)^k (1+\beta)^{n-k}}{2^n} R(\mathrm{x}), \qquad \text{(A8)}$$

where $R(\mathrm{x}) = 1 + (\gamma+\beta)[(\sum_{l:x_l=1} p_l)/(1-\beta) - (\sum_{m:x_m=0} p_m)/(1+\beta)]$. The posterior probabilities follow.

$$
\begin{aligned}
p_i(\mathrm{x} : x_i = 1) &= \frac{(1+\gamma)p_i}{(1-\beta)R(\mathrm{x})}; \\
p_i(\mathrm{x} : x_i = 0) &= \frac{(1-\gamma)p_i}{(1+\beta)R(\mathrm{x})}; \qquad \text{(A9)} \\
p_j(\mathrm{x} : j \text{ not tested}) &= \frac{p_j}{R(\mathrm{x})}. \qquad \text{(A10)}
\end{aligned}
$$

Denote with $\mathrm{x}(k)$ a profile with k positive signals. There are $\binom{n}{k}$ different such profiles. The posterior entropy from testing n candidates is:

$$
\begin{aligned}
H_{post}(n) = -\sum_{k=0}^{n} & \frac{(1-\beta)^k (1+\beta)^{n-k}}{2^n} \\
& \times \Bigg\{ \sum_{i=1}^{n} \Bigg[\sum_{\mathrm{x}(k):x_i=1} \frac{1+\gamma}{1-\beta} p_i \log\left[\frac{(1+\gamma)p_i}{(1-\beta)R(\mathrm{x}(k))}\right] \\
& \quad + \sum_{\mathrm{x}(k):x_i=0} \frac{1-\gamma}{1+\beta} p_i \log\left[\frac{(1-\gamma)p_i}{(1+\beta)R(\mathrm{x}(k))}\right]\Bigg] \\
& \quad + \sum_j \sum_{\mathrm{x}(k)} p_j \log\left[\frac{p_j}{R(\mathrm{x}(k)}\right]\Bigg\}. \qquad \text{(A11)}
\end{aligned}
$$

Because of the independence of the individual test outcomes, it is optimal for the $(l+1)$st test candidate, given that $l < n$ candidates are already chosen, to be closest to p^* (among the remaining candidates). As we have seen in Theorem 2a that p^* always implies that one of the two most likely candidates to be chosen first, and the second only when both are close to $1/2$, it is optimal to test the n most likely candidates when $n \ge 2$.

So far, we have taken n as given. Now consider the dynamic program of the entropy proxy problem, given the optimal policy

for any $n: V(H) = \min_n[nc + c_\tau + V(H_{post}(n))]$ holding n constant at the chosen value from now on. Observe that a larger γ and β each by itself reduces the posterior entropy $H_{post}(n)$ (both spread the arguments of the log functions). In addition, a larger number of parallel tests decreases $H_{post}(n)$ convexly. We can show that $\frac{1}{2}[H_{post}(n) + H_{post}(n+2)] > H_{post}(n+1)$. The proofs of these statements are messy and omitted here (they can be obtained from the authors). We can show that $V(H_{post}(n))$ is increasing in $H_{post}(n)$. Thus, convexity of $H_{post}(n)$ together with the linear direct cost cn implies that there is a unique n^*. When we approximate $H_{post}(n)$ by a continuous function in n, the implicit function theorem implies

$$\frac{\partial n^*}{\partial \gamma} = -\frac{\partial^2 H_{post}(n)/(\partial\gamma\partial n)}{\partial^2 H_n/\partial n^2} \geq 0,$$

and the same holds for β. n^* increases weakly in the fidelities because it is integer.

Finally, as in Theorem 2a, this result holds for *any* target entropy H_0. We can thus condition on any future state $\mathbf{p}$ (for example, the next time we want to change n), set H_0 as the corresponding entropy level, and apply the optimal n until then. Thus, the theorem holds also for the original dynamic program. This proves the theorem. □

Proof of Theorem 3. We first calculate an upper limit on C_{mod}. As the M independent PEs can be tested in parallel, the costs of the tests simply add up. The time to test each PE is a random variable that can vary between 1 (first batch contains the solution) and $n(N)$ (last batch contains the solution). The expected time to test M PEs in parallel is the expectation of the maximum of these random variables. The expectation of the maximum of M independent uniformly distributed random variables is $(M/(M+1))n$. From Corollary 1, the testing time distribution is skewed to the left: Expected testing time is $(n(N)+1)/3$. Thus, the expectation of the maximum is smaller than for a uniform distribution. The test costs simply add up for the M PEs. This gives the bound on the total cost in the middle column. The extreme cases for parallel and sequential testing (left and right columns) follow directly from Corollary 1.

For estimating C_{sequ}, assume first that the M PEs are tested sequentially, upstream before downstream. Then the total costs simply add up, both in time and in the number of tests, which gives the middle row of the theorem. Columns 1 and 3 are trivially larger than the corresponding C_{mod}. The middle column is larger than C_{mod} for any n because $n/(M+1) < (n+1)/3$. It may be possible to reduce C_{sequ} by testing an upstream and a downstream PE in an overlapped manner. The best that can be achieved by overlapping is C_{mod}, provided that downstream picks the correct upstream alternative as the assumed solution and tests only its own alternatives compatible with this assumed upstream solution. The overlapped cost is larger than C_{mod} in expectation. This proves the comparison statement in Corollary 2.

Finally, we estimate C_{int}. In the integral case, the solution of one PE depends on the solutions of the others, and therefore, all combinations of alternatives must be tested. This is equivalent to one PE with N^M alternatives. This gives the third row of the theorem. The conditions for the extreme cases (parallel or sequential testing) change because the number of alternatives is now different; a PE of N candidates may be tested sequentially, while it may be optimal to test partially in parallel in the PE of N^M candidates.

Inspection shows that for $2cN^M/c_\tau$ large, $C_{int} > C_{sequ}$. Numerical analyses show that $C_{int} > C_{sequ}$ for all possible parameter constellations as long as $3/8N \leq c/c_\tau$ holds (see Corollary 1). When delay costs are so high that this condition is not fulfilled, tests are performed in parallel (Corollary 1), and the total costs of testing multiple PEs become the same in both cases.[1] Again, C_{mod} is smallest, and $C_{int} > C_{sequ}$ iff $c/c_\tau > (M-1)/(N(N^{M-1}-M))$. If c/c_τ is even smaller, it is optimal to test sequentially dependent PEs in parallel, incurring the extra cost of testing all combinations of alternatives in order to gain time. In this extreme case, $C_{int} = C_{sequ}$. This proves Theorem 3 and Corollary 2. □

References

Abernathy, W., R. Rosenbloom. 1968. Parallel and sequential R&D strategies: Application of a simple model. *IEEE Trans. Engrg. Management* **15**(1) 2–10.

Alchian, A. 1950. Uncertainty, evolution and economic theory. *J. Political Econom.* **58**(3) 211–221.

Alexander, C. 1964. *Notes on the Synthesis of Form.* Harvard University Press, Cambridge, MA.

Allen, T.J. 1966. Studies of the problem-solving process in engineering design. *IEEE Trans Engrg. Management* **EM-13**(2) 72–83.

——. 1977. *Managing the Flow of Technology*. MIT Press, Cambridge, MA.

Baldwin, C., K. Clark. 1997a. Design options and design evolution. Working paper 97-038, Harvard Business School, Boston, MA.

——, ——. 1997b. The value of modularity: Splitting and substitution. Working paper 97-039, Harvard Business School, Cambridge, MA.

Bertsekas, D.P. 1995. *Dynamic Programming and Optimal Control.* Athena Scientific, Belmont, MA.

Bohn, R.E. 1987. Learning by experimentation in manufacturing. Working paper No. 88-001, Harvard Business School, Boston, MA.

——. 1995. Noise and learning in semiconductor manufacturing. *Management Sci.* **41**(1) 31–42.

Clark, K.B. 1985. The interaction of design hierarchies and market concepts in technological evolution. *Res. Policy* **14** 235–251.

——, T. Fujimoto. 1989. Lead time in automobile development: Explaining the Japanese advantage. *J. Tech. Engrg. Management* **6** 25–58.

Cusumano, M., R. Selby. 1995. *Microsoft Secrets.* The Free Press, New York.

Dahan, E. 1998. Parallel and sequential prototyping in product development. Unpublished Ph.D. dissertation, Stanford University, Stanford, CA.

[1] Here, we assume that $3/8N^f > c/c_t$ also holds. If not, the integral design will not be tested fully in parallel, which makes the argument slightly more complicated (omitted here).

DeGroot, M.H. 1970. *Optimal Statistical Decisions.* McGraw-Hill, New York.

Eisenhardt, K.M., B.N. Tabrizi. 1995. Accelerating adaptive processes: Product innovation in the global computer industry. *Admin. Sci. Quart.* **40**(1) 84–110.

Eppinger, S.D., D.E. Whitney, R.P. Smith, D.A. Gebala. 1994. A model-based method for organizing tasks in product development. *Res. Engrg. Design* **6**(1) 1–13.

Huberman, B.A., T. Hogg. 1988. The behavior of computational ecologies. B.A. Huberman, ed. *The Ecology of Computation.* North Holland–Elsevier, 77–115.

Iansiti, M. 2000. How the incumbent can win: Managing technological transitions in the semiconductor industry. *Management Sci.* **41**(2) 169–185.

Kauffman, S., S. Levin. 1987. Towards a general theory of adaptive walks on rugged landscapes. *J. Theoret. Biology* **128** 11–45.

Loch, C.H., C. Terwiesch. 1998. Communication and uncertainty in concurrent engineering. *Management Sci.* **44**(8) 1032–1048.

Marples. D.L. 1961. The decisions of engineering design. IRE *Trans. Engrg. Management* Vol. **EM-8**, 55–71.

Montgomery, D. 1991. *Design and Analysis of Experiments.* Wiley, New York.

Nelson, R. 1961. Uncertainty, learning, and the economics of parallel research and development efforts. *Rev. Econom. Statist.* **43** 351–364.

Quinn, M. 1987. *Designing Efficient Algorithms for Parallel Computers.* McGraw-Hill, New York.

Reinertsen, D. 1997. *Managing the Design Factory.* The Free Press, New York.

Shannon, C.E. 1948. A mathematical theory of communication. *Bell Systems Tech. J.* **27** 379–423 and 623–656.

Simon, H.A. 1969. *The Sciences of the Artificial* 2nd ed. (1981). MIT Press, Cambridge, MA.

Smith, R.P., S.D. Eppinger. 1997. A predictive model of sequential iteration in engineering design. *Management Sci.* **43** 1104–1120.

Sobek, D.K., A.C. Ward, J.K. Liker. 1999. Toyota's principles of set-based concurrent engineering. *Sloan Management Rev.* **40**(2) 67–83.

Steward, D.V. 1981. *Systems Analysis and Management: Structure, Strategy, and Design.* Petrocelli Books, New York.

Stone, L.D. 1975. *Theory of Optimal Search* Vol. 118. Mathematics in Science and Engineering, Academic Press.

Suh, N.P. 1990. *The Principles of Design.* Oxford University Press, Oxford, U.K.

Terwiesch, C., C.H. Loch. 1999. Measuring the effectiveness of overlapping development activities. *Management Sci.* **45**(4) 455–465.

——, ——, A. De Meyer. 1999. Exchanging preliminary information in concurrent development processes. Working paper, Wharton/INSEAD.

Thomke, S. 1998. Managing experimentation in the design of new products. *Management Sci.* **44** 743–762.

——, D. Bell. 2001. Optimal testing in product development. *Management Sci.* **47** Forthcoming.

——, E. von Hippel, R. Franke. 1998. Modes of experimentation: An innovation process—and competitive—variable. *Res. Policy* **27** 315–332.

Ulrich, K. 1995. The role of product architecture in the manufacturing firm. *Res. Policy* **24** 419–440.

——, S. Eppinger. 2000. *Product Design and Development*, 2nd ed. McGraw-Hill, New York.

Weitzman, M.L. 1979. Optimal search for the best alternative. *Econometrica* **47** 641–654.

Wheelwright, S.C., K.B. Clark. 1992. *Revolutionizing Product Development*. The Free Press, New York.

Accepted by Hau Lee; received October 4, 1999. This paper was with the authors 3 months for 1 revision.