



# The Effect of “Front-Loading” Problem-Solving on Product Development Performance

Stefan Thomke and Takahiro Fujimoto

*In recent years, there has been a growing interest in the link between problem-solving capabilities and product development performance. In this article, the authors apply a problem-solving perspective to the management of product development and suggest how shifting the identification and solving of problems—a concept that they define as front-loading—can reduce development time and cost and thus free up resources to be more innovative in the marketplace.*

*The authors develop a framework of front-loading problem-solving and present related examples and case evidence from development practice. These examples include Boeing’s and Chrysler’s experience with the use of “digital mock-ups” to identify interference problems that are very costly to solve if identified further downstream—sometimes as late as during or—after first full-scale assembly.*

*In the article, the authors propose that front-loading can be achieved using a number of different approaches, two of which are discussed in detail: (1) project-to-project knowledge transfer—leverage previous projects by transferring problem and solution-specific information to new projects; and (2) rapid problem-solving—leverage advanced technologies and methods to increase the overall rate at which development problems are identified and solved. Methods for improving project-to-project knowledge transfer include the effective use of “post-mortems,” which are records of post-project learning and thus can be instrumental in carrying forward the knowledge from current and past projects. As the article suggests, rapid problem-solving can be achieved by optimally combining new technologies (such as computer simulation) that allow for faster problem-solving cycles with traditional technologies (such as late stage prototypes), which usually provide higher fidelity.*

*A field study of front-loading at Toyota Motor Corporation shows how a systematic effort to front-load its development process has, in effect, shifted problem-identification and problem-solving to earlier stages of product development. They conclude the article with a discussion of other approaches to front-load problem-solving in product development and propose how a problem-solving perspective can help managers to build capabilities for higher development performance. © 2000 Elsevier Science Inc.*

Address correspondence to Takahiro Fujimoto, University of Tokyo, Faculty of Economics, Tokyo 113, Japan or Stefan Thomke, Harvard Business School, Morgan Hall T63, Boston, MA 02163.

**BIOGRAPHICAL SKETCHES**

**Stefan Thomke** is Assistant Professor in the Technology and Operations Management Area at the Harvard Business School. His research focuses on managing flexibility, experimentation strategies, and learning in the product and technological innovation process. His writings have been published in several books and journals, including *Research Policy*, *Management Science*, *California Management Review*, and *Scientific American*. Thomke holds B.S. and M.S. degrees in Electrical Engineering, a S.M. degree in Operations Research, a S.M. degree in Management from the MIT Sloan School, and a Ph.D. degree in Electrical Engineering and Management from the Massachusetts Institute of Technology (MIT), where he was awarded a Lemelson-MIT doctoral fellowship for invention and innovation research. His doctoral research won first prize in the Product Development & Management Association's (PDMA) 1994 International Dissertation Proposal Competition.

**Takahiro (Taka) Fujimoto** is Professor at Tokyo University's Faculty of Economics and Senior Research Associate at the Harvard Business School. His research interests are very broad, but have focused primarily on product development and manufacturing systems in the world automotive industry. He has written extensively on the management of product development and innovation, including the 1991 book *Product Development Performance: Strategy, Organization, and Management in the World Auto Industry* (with Kim Clark). His latest book, *The Evolution of a Manufacturing System at Toyota*, will be available in 1999 (Oxford University Press). Fujimoto is a graduate of Tokyo University and received his Doctor of Business Administration (DBA) from the Harvard Business School where he was also a research associate until his appointment to the faculty of Tokyo University.

*“Managing the problem solving curve is vital to our company’s survival and front-loading is one of the main arenas of our current capability-building.”*

Managing Director at Leading Automotive Firm (1998)

**Introduction**

The link between problem-solving capabilities and superior product development performance has become the subject of a growing body of research in recent years [8,24,25,36,38,44,46]. For example, shorter development times—an important dimension of development performance—have been associated with overlapping problem-solving activities [8,21,22,35]. In this article, we propose a direct link between *early* problem-solving and development performance and discuss different approaches that firms can use to improve their development processes.

To understand how problem-solving and performance are related, we adopt a problem-solving perspective that increasingly is recognized by researchers as being fundamental to product development. Man-

agers of development projects usually are very concerned about the identification of problems, because solving them becomes, on average, increasingly expensive and time-consuming as projects progress and financial commitments are made. With the recent emergence of new technologies and methods—such as computer-aided engineering (CAE) and rapid prototyping—that aid in the acceleration of problem-solving, it is not surprising that some practitioners have initiated concentrated efforts to reengineer their development processes to move (or “load”) their problem identification and solution backward in time (to the “front” of the process).<sup>1</sup> In this article, we provide a conceptual framework and some evidence from development practice to describe the basic principle behind “front-loading.”

We base our discussion on a problem-solving perspective of product development. We describe and analyze a product development project as a bundle of interdependent problem-solving cycles that include modeling and testing via computers or physical prototypes as core activities. Using this perspective, we define front-loading problem-solving as *a strategy that seeks to improve development performance by shifting the identification and solving of [design] problems to earlier phases of a product development process*. While we present examples of front-loading from different industries, we primarily report on findings from the automotive sector, where the methodology currently is being applied to the reengineering and shortening of product development. Our evidence is mostly case-based, but we feel it points toward the emergence of an important capability for improving development practice.

The article proceeds as follows. We start with a very brief discussion of development lead time. We then describe our problem-solving perspective of product development and apply it to automotive development, where much of our empirical evidence was collected. In the next sections we explain the basic principle behind front-loading, including several examples from current practice, which is followed by a field study of front-loading at Toyota Motor Corporation. We conclude the article with a discussion of other applications that may result from front-loading, such as its role in the building of development capabilities.

<sup>1</sup>For example, the automotive firm BMW recently initiated a large-scale reengineering program that aims to reduce development time of new models by 50%. New technologies such as “digital mock-ups” and advanced computer simulation play an integral role in these efforts [40].

## Product Development Lead Time

The performance of development processes generally can be measured along multiple dimensions such as lead time, productivity, and product quality [9]. Development lead time is a measure of how quickly a firm can move a product from concept to market, whereas productivity relates to the level of resources (e.g., engineering hours, material, equipment) required to accomplish the same objective. The output of a process, however, is a product, and its complexity and the extent to which it conforms to customer expectations drive product quality. These three performance dimensions are closely related—any attempts to change one variable can have consequences for the other two in ways that sometimes are difficult to predict.

In the remainder of the article, we primarily focus on lead time. The extensive literature on product development speed reports on a number of different approaches or characteristics associated with shorter development times, including the level of human resources [6], overlapping tasks [7,8,21,29,35], cross-functional teams, and pre-development work [9,10,46], and the utilization of carry-over parts and platforms [9,12,26,31]. Discussions of the different factors and strategies to reduce development time can be found in Crawford [11], Griffin [16]; Table 2], Gupta and Wilemon [17], Smith and Reinertsen [34], Wheelwright and Clark [46], and others. The objective of this article is not to add another factor or approach to the myriad of studies already published on development time. Instead, we propose that both academics and practitioners can benefit from examining product development processes and its link to performance through the lens of problem-solving. More specifically, we discuss different approaches for early problem-solving—grouped under the concept of “front-loading”—and examine how they can affect development performance.

## A Problem-Solving Perspective of Product Development

Whereas the early innovation and R&D management literature views problem-solving as a fundamental design activity [4,23,32,33], only in recent years has problem-solving been discussed and explicitly adopted in the product development literature [8,14,24,36,38,41,46]. In this article, we view problem-solving as an iterative process, driven by trial-and-error

experiments that are guided by knowledge of underlying relationships between cause and effect (Figure 1).

Problem-solving starts with problem recognition and goal definition and continues with an iterative process of experimental search through alternatives that are designed and built during step 1 (design) and step 2 (build models) of a four-step problem-solving cycle.<sup>2</sup> These alternatives may or may not include the best possible solutions—one has no way of knowing. The alternatives are tested against an array of requirements and constraints during step 3 (run experiments). Test outcomes are analyzed during step 4 (analyze and evaluate) and used to revise and refine the solutions under development, and progress is made in this way toward an acceptable result. For example, one might (step 1) conceive of and design a new, more rapidly deploying airbag for a car; (step 2) build a prototype of key elements of that airbag as well as any special apparatus needed to test its speed of deployment; (step 3) run the experiment to determine actual deployment speed; and (step 4) analyze the result. If the results of a first experiment are satisfactory, one stops after step 4. However, if, as is usually the case, analysis shows that the results of the initial test(s) are not satisfactory, one may elect to modify one’s experiment and “iterate” (or try) again.

Modifications may involve the experimental design, experimental conditions, fidelity of the experimental setup, or even the nature of the desired solution. The new information provided by a problem-solving cycle to a designer are those aspects of the outcome that he or she did not (was not able to) know or foresee or predict in advance—the “problem” or “error” [3,28].

Applying the problem-solving perspective to automotive development (Figure 2), one finds that it consists of a bundle of numerous *problem-solving cycles*, each of which consists of design, build, run, and analysis activities. Problem-solving cycles can be small, involving only a single designer (such as individual simulation experiments) or they can be very large, involving several development groups (such as major prototyping cycles). As projects progress, cycles tend to include models of increasing completeness, or

<sup>2</sup>Similar building blocks to analyze the design and development process were used in earlier research. Simon [33] examined design as series of “generator-test cycles.” Clark and Fujimoto [8] and Wheelwright and Clark ([46]; Chapters 9 and 10) used “design-build-test” cycles as a framework for problem-solving in product development. Thomke [36] modified the blocks to include “run” and “analyze” as two explicit steps that conceptually separate the execution of an experiment and the learning that takes place during analysis.

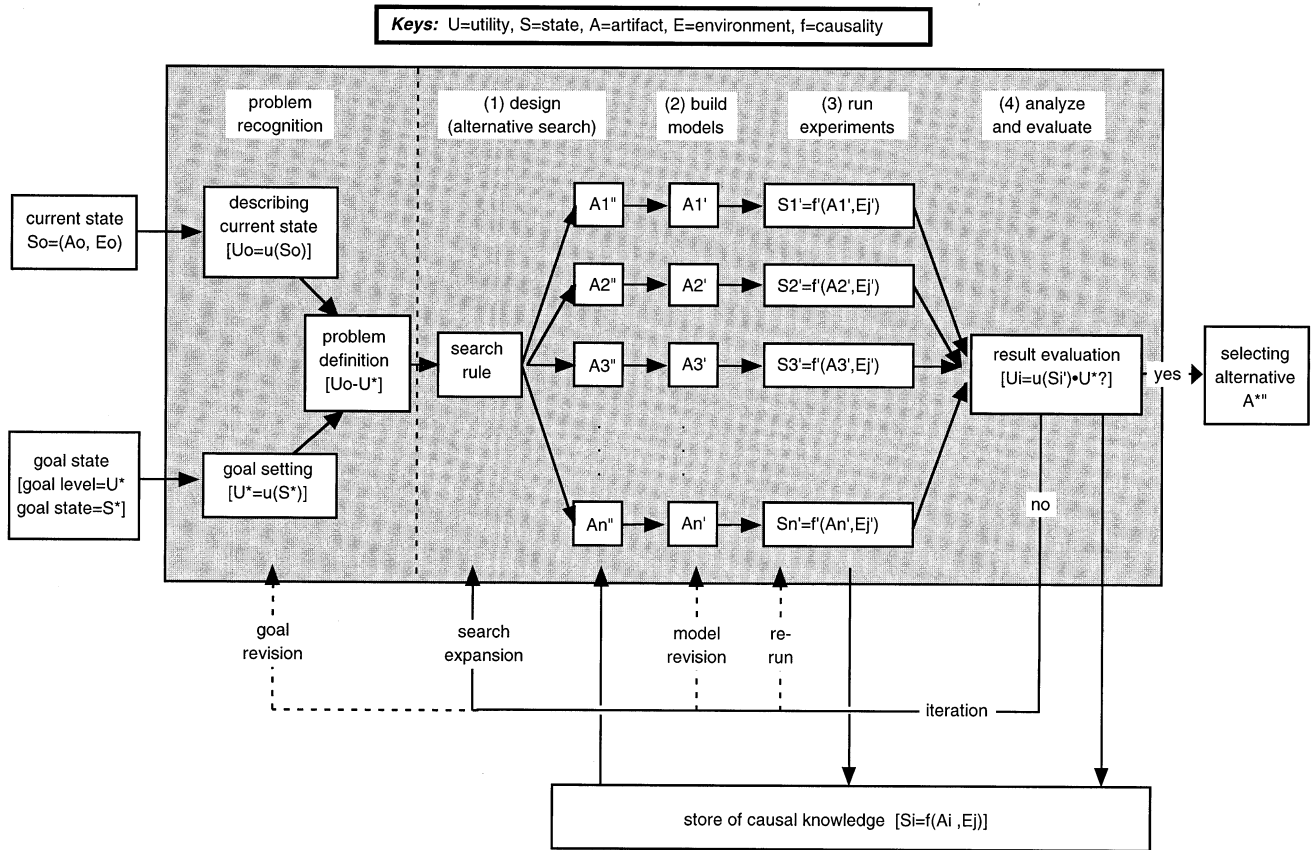


Figure 1. Product development as a process of repeated problem-solving [9].

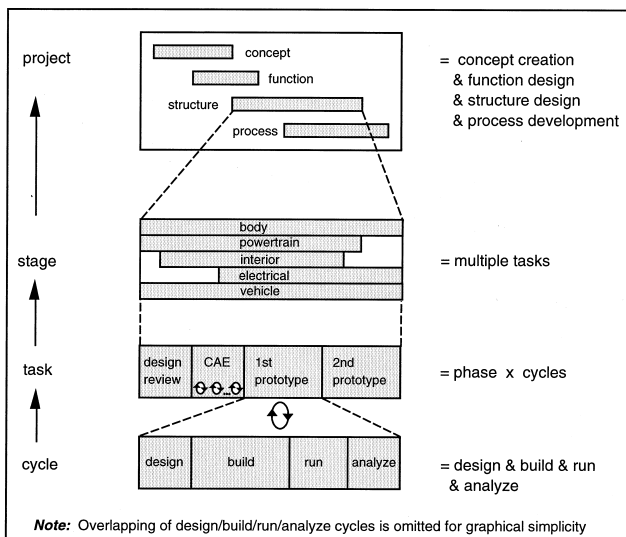


Figure 2. Automotive development as problem-solving cycles and stages.

geometric fit, and manufacturability [14]. Sometimes a model is incomplete because one cannot economically incorporate all respects of “reality” (i.e., the *full-fidelity* model being used under *real* conditions) or simply does not know them. Other times, it is economical to build incomplete models in order to reduce investments in aspects of “reality” that are irrelevant to the problem being solved. Thus, a model of an airplane tested in wind tunnel experiments has no internal design details—these are both costly to model and mostly irrelevant to the aerodynamics problem being solved. Problem-solving cycles can be structured in hierarchical form: they are iterated to complete a *task* that creates a solution for each component; tasks then are integrated into major *stages* of development such as product engineering and process engineering.

### Early Problem-Solving Through “Front-Loading”

fidelity, (e.g., thought experiments, computer simulations, physical prototypes, pilot vehicles, etc.) for testing the effect of design decisions on functionality,

An important part of an effective development strategy is the timing and fidelity of test models [39,46]. The information generated from these models plays an

integral role in the identification and solution of design and manufacturing problems. If models are built and tested very late in a development process, the cost and time required to solve identified problems can be very large. Thus, it is not surprising that the benefits of early problem identification and solving can be quite remarkable and provide an area of great leverage for improving product development performance. For example, a study of several large software projects showed the relative cost of correcting software errors (or making software changes) to go up significantly as a function of the phase in which the corrections or changes were made [5]. It was found that a software requirements error corrected during the early specification phase consisted merely of updating the requirement specifications. A correction in the (very late) maintenance phase, however, involved a much larger inventory of specifications, code, user and maintenance manuals, training material, and, of course, re-validation. On average, the study found a change in the maintenance phase to be roughly 100 times more costly than in the specification phase, not counting any indirect operational problems in the field. Whereas considerable progress to make development processes more flexible has been made in recent years [18,37], late engineering changes as a response to identified design problems still can be very costly and time consuming. This is particularly true in automotive development where late design changes can cost millions of dollars and take weeks or months to be carried out—partially due to increasing tooling commitments.

#### *Definition of Front-Loading Problem-Solving*

We define front-loading problem-solving as a *strategy that seeks to improve development performance by shifting the identification and solving of [design] problems to earlier phases of a product development process*. We propose that effective front-loading can be achieved using a number of different approaches, two of which will be presented in detail. These two approaches have been proven to be very important but, in our opinion, have not received the kind of academic attention they merit.

The first approach, *project-to-project knowledge transfer*, involves the effective transfer of problem- and solution-specific information between development projects to reduce the total number of problems to be solved from the onset of development activities. For example, postmortem reports prepared by software developers after a project is complete usually

include detailed information on problems (or “bugs”) that can be reviewed by teams prior to the start of a new project.

The second approach, *rapid problem-solving*, leverages advanced technologies and methods to increase the overall rate at which development problems are identified and solved. Consider, for example, that building prototype vehicles that can be used in a crash test can take many months and thus limits the rate at which crashworthiness-related problems can be identified. However, the availability of lower cost and faster computer-aided engineering (CAE) tools permit higher rates of problem-solving, particularly at the early phases of product development. Combined with traditional hardware tests, these advanced technologies permit significantly faster and more frequent problem-solving cycles.

#### *Front-Loading Problem-Solving: A Taxonomy and a Conceptual Framework*

To solve a problem, problem- and solution-related information must be created, made available, and recognized by the problem-solver. Studies of the problem identification process show that such information can become available to a developer in two ways: (1) it already exists as very similar problems were identified and solved in prior development projects; and (2) it is created as part of repeated problem-solving during the development process [43].

With respect to problems involving information that already exists, a process should be in place that allows problem identification and solving at the start of the project. However, firms often neglect project-to-project learning and information transfer, resulting in the “rediscovery” of old problems in new projects. Or developers are simply unable to cope with the complexities of large development projects: the information generally is available to them but they are unable to predict the often subtle chain of cause and effect that leads to a design problem. In their study of 27 field problems that affected two novel process machines, von Hippel and Tyre [43] found that 15 of 22 problems identified after the machines were installed in the field involved information that existed prior to the installation. In 10 of the 15 problems, the problem-specific information was not transferred between designers; in the remaining five problems, the designers had the information but were unable to use it effectively.

With respect to problems involving information that is created as part of the problem-solving process, it is

desirable to create such information as early as possible. One could, for example, increase early problem-solving capacity by shifting ample design resources and increased testing budgets to the very early stages of a new development project. However, development practice often looks different: resources are ramped up slowly as a project unfolds, and thus problem-solving activities and the related generation of information ramp up slowly as well. Hence, observed patterns of problem identification often are shifted downstream (“end-loaded”). The objective of this article is to present the concept of front-loading as a way to reverse this effect and to shift problem-identification and problem-solving upstream.

To conceptualize front-loading with the aid of a framework, consider a product development process where problems are identified and solved in an iterative fashion, following the design-build-run-analyze (d-b-r-a) cycles described earlier (Figure 3). Suppose that we develop in an environment where problems are identified via high-fidelity (*h-f*) prototype testing only and no problem or solution-specific information is being transferred between projects (Figure 3; case a). Problems are identified and solved at a constant rate  $r_{h-f}$  that depends on the speed at which a d-b-r-a cycle can be completed.<sup>3</sup> Total development time  $T$  can only be shortened if the problem-solving rate  $r_{h-f}$  is increased by carrying out d-b-r-a cycles more rapidly. These cycles can be *compressed* by restructuring prototype build and test processes (e.g., by adding capacity to bottleneck operations) or through a change in the

explicit or implicit incentive structure such that early problem-solving is emphasized.

However, if a second class of test prototypes (e.g., “virtual” prototypes generated by a computer) of lower fidelity (*l-f*) but a higher problem-solving rate  $r_{l-f}$  was available and knowledge could be transferred between projects (Figure 3; case b), developers would be able to use the following two approaches to shift their problem-solving trajectory:

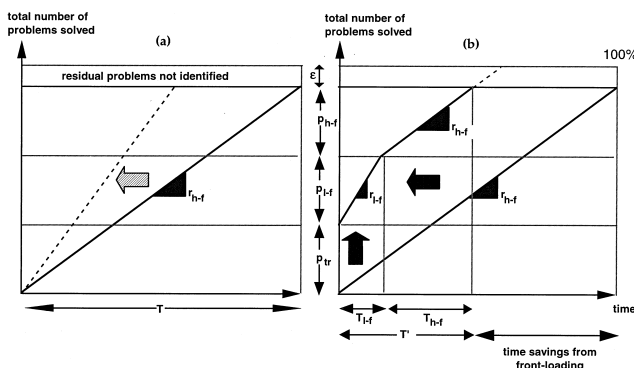
- Increase the initial number of development problems solved, or “avoided,” by more effective project-to-project transfer of problem and solution-specific information ( $p_{tr}$ ).
- Use the lower-fidelity prototype to solve development problems that it can identify ( $p_{l-f}$ ) more rapidly and then *switch* to the slower higher fidelity prototype for the remaining problems ( $p_{h-f}$ ).

The combined benefit will be a shorter total development time  $T$  as shown in Figure 3, case b. In general, management could extend the same logic to determine an optimal number of prototypes  $n$ , each with a different fidelity, such that total development time and cost is minimized. The price one would pay for a large  $n$  is the cost of repeated testing via multiple prototypes and the benefit would be the availability of early information. (For a formal treatment of this important tension, see Thomke and Bell [39]). The opportunities that advanced technologies, such as computer simulation, three-dimensional computer-aided design (CAD), and rapid prototyping, can provide now become quite apparent: even though they sometimes are of lower fidelity than full physical prototypes, they can identify a significant percentage of total development problems at a rate significantly higher than conventional high-fidelity prototypes. Combined with traditional hardware prototyping, we suggest that these technologies can result in remarkable development time reductions through earlier and more rapid problem-solving.

### Managing Front-Loading: Evidence from Industrial Practice

In the previous section, we used a problem-solving perspective to identify two general approaches to the management of front-loading activities—activities that would result in a shift of problem-identification and problem-solving to earlier phases of product development. The first approach (*project-to-project knowledge transfer*) involves the transfer of problem- and solution-specific information between development

<sup>3</sup>To keep the framework at a conceptual level, we are using a linear approximation of problem-solving rates. Thomke [36] showed that this rate is more likely to experience diminishing returns and thus follow a nonlinear trajectory.



**Figure 3.** Problem-solving cycles with (a) a single high-fidelity (*h-f*) mode and no project-to-project knowledge transfer; and (b) two modes (*l-f* and *h-f*) with knowledge transfer (for simplicity, trajectories are shown to be linear).

projects to reduce the total number of problems to be solved from the onset of development activities. The second approach (*rapid problem-solving*) leverages advanced technologies and methods to increase the overall rate at which development problems are identified and solved. As Figure 3 shows, each of these approaches can result in an upstream shift of problem-solving, and thus their combination enable firms to reduce development time and cost significantly. We now present examples from development practice that demonstrate the benefits of both approaches.

### *Project-to-Project Knowledge Transfer*

Studies of problem-solving have shown that firms often find old problems in new development projects. For example, in their study of the development of front and rear auto body closures (i.e., hoods, trunk lids, and lift gates), Watkins and Clark [45] found that design problems often were repeated between consecutive projects. (As an example, they presented one problem that showed up repeatedly over three sequential projects.) As discussed earlier, von Hippel and Tyre [43] observed a similar pattern and found that not only was there a lack of problem-specific information being transferred, but designers sometimes were unable to use the transferred information effectively. Thus, it appears that more effective transfer of knowledge between projects can improve development performance [2,12]. This observation is shared by Adler [1], who studied different coordination mechanisms and interdependence issues in the design and manufacturing of printed circuit boards for electronic components and hydraulic tubing for aircraft. Among three different phases of coordination, he found pre-project [coordination] activities to have a key role in capturing the learning from previous projects and in the acceleration of new product development. In this article, we are particularly interested in one kind of pre-project coordination mechanism: the transfer of problem- and solution-specific information that can reduce the overall number of problems for a new project, as shown in Figure 3.

As an example of effective transfer practice, consider the use of “postmortems” in the development of computer software. Good postmortems are detailed records of a project’s history and include, among other things, information on specific product and process problems discovered at various stages of software development. Cusumano and Selby [13] reported that much of the learning between projects at Microsoft

can be attributed to its systematic use of such postmortem reports. In their research, they found that development teams generally take 3 to 6 months to prepare a postmortem, which can be between less than 10 to more than 100 pages long. In addition to accounting for people, product, and scheduling issues, the postmortems also contain detailed information on number of problems [bugs] identified, problem severity, and record of finding and solving problems. Preparing, discussing, and reviewing these postmortem reports, particularly before and/or at the beginning of a new project, has proven to be instrumental in carrying forward the knowledge from current and past projects. In their many years of using postmortems, Microsoft also discovered that transferring information on problems alone is very helpful but not sufficient; they needed to understand why a problem occurred and what solutions are possible [13]. Equipped with such information, developers can move more quickly toward the early identification and solution of problems that seemed novel at first but were experienced in different forms during past projects.

Postmortems are a very good example of effective transfer mechanism when the information being transferred can be economically encoded; however, that may not always be the case. In a study of 229 project members of 25 Japanese automobile projects, Aoshima [2] found that effective knowledge transfer mechanisms were also a function of the kind of knowledge being transferred. When information can be easily encoded, such as component-level data, he found that it was more effective to use archival-based mechanisms (documents, drawings). However, if the knowledge is “sticky” [42] or very costly to encode and transfer, such as tacit knowledge about the integration of components, firms performed better if they relied on face-to-face communication, people transfers, and other mechanisms that allow for effective transfer of such tacit knowledge between projects.

### *Rapid Problem-Solving Using Advanced Technologies and Methods*

In this second approach, we are primarily concerned with the rate at which new problems are being solved. To gain deeper insights into the kinds of technologies and methods that are available to accelerate problem-solving, we divide problems into two categories—fit and function.

When different parts and/or subsystems occupy the same coordinates in three-dimensional geometric

space, they interfere with each other, that is, they do not fit. Such problems are known as “interference” problems and are very common during the geometric integration of a complex product. Problems of fit are different than problems of function, where developers are concerned with the actual performance of a product (such as the fuel consumption or aerodynamics of an airplane). With the emergence of advanced computer technologies, many companies have been able to complement traditional hardware-based models with so-called “virtual” approaches utilizing computer models.<sup>4</sup> However, as we will see, the technologies and process changes chosen depend on the problems being solved.

#### *Early Identification of Problems of Fit Through Digital Mock-Ups*

Designers often are unaware that their designs interfere in space, and the traditional approach of mapping three-dimensional designs onto separate two-dimensional drawings provides very limited help in the identification of such interference problems. Complex products can involve thousands or hundreds of thousands of parts that could potentially interfere in three-dimensional space—designed by engineers who often do not even know each other.

When Boeing developed its new 777 aircraft, they also designed a new process for problem identification and correction [30].<sup>5</sup> Experience had taught them that their prior use of increasingly refined physical prototypes (also known as mock-ups) detected most design problems—but not all of them—and many of them very late in the development process. Developers were particularly concerned with interference problems, as the account of one Boeing chief engineer tells us:

“You have five thousand engineers designing the airplane. It’s very difficult for those engineers to coordinate with two-dimensional pieces of paper, or for a designer who is designing an air-conditioning duct to walk over to somebody who is in Structures and say, ‘Now, here’s my duct—how does it match with your structure?’ Very difficult with two-dimensional pictures. So we ended up using the [physical] mock-up and, quite honestly, also using the final assembly line

to finish up the integration. And it’s very costly. You end up with an airplane that’s very difficult to build. The first time that parts come together is on the assembly line. And they don’t fit. So we have a tremendous cost on the first few airplanes of reworking to make sure that all the parts fit together.” (Sabagh [30])

Boeing’s management wanted a process that allowed them to “front-load” [our words] interference problem-identification and correction to earlier points of the 777 development. They took advantage of the three-dimensional CAD system CATIA® and coupled it with an in-house software that enabled engineers to assemble and test “digital mock-ups” for interference problems. Similar to physical mock-ups that are built to detect problems of fit, “digital mock-ups” allow a virtual assembly of a product that can be checked automatically for interferences. While these automatic interference checks could be performed many times throughout the development process, they also changed the way people interacted with each other. Designers were more likely to change their designs early and relied on others to track these changes using the new computer technology. To add discipline and structure to the problem-solving process, Boeing instituted a process that was divided into alternating periods of design and stabilization. During the design period, engineers were allowed to make changes. During the stabilize period, a software checked for interference problems that had to be resolved before proceeding to the next design period. The resolution of interference problems was no trivial task as shown by an early interference test of twenty pieces of the 777 flap (wing section): the software made 207,601 checks that resulted in 251 interference problems—problems that would have been very costly and time-consuming to correct during final assembly.<sup>6</sup>

Other firms currently are experiencing similar benefits from using advanced technology to front-load problem-solving. Chrysler Corporation discovered

<sup>4</sup>Adler [1] made a similar observation. He found that computer-aided technologies (CAD/CAM) not only helped organizations to avoid errors that would normally result in costly engineering change orders, but also allowed them to process these changes or “problem solutions” more rapidly.

<sup>5</sup>The description of Boeing’s development practice is based on Sabagh’s [30] detailed account of the 777 aircraft project.

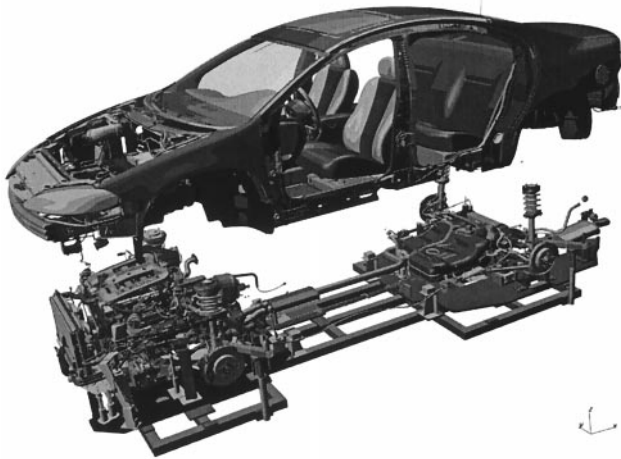
<sup>6</sup>We also should note that the use of interference testing via three-dimensional CAD led to unusual but very productive interactions between design engineers. For example, Alan Mulally, director of engineering on the 777, reported on an incident where he saw a senior structures engineer going up and down the building looking for a hydraulic designer. The engineer wanted to put a bracket on his floor beam, and he and the hydraulic designer had not come to an agreement on the location and the size of the bracket and whether it was going to create an interference. Extremely mad, the engineer stopped Mulally and asked “what they [the hydraulic designers] looked like. Do they have tubes in their pockets? Do they have tubes coming out of their heads?” (Sabagh [30]). The unusual interaction arose because interference testing via three-dimensional CAD forced designers to interact and solve integration problems early.



that the use of three-dimensional CAD mock-ups (internally known as digital mock-ups) could help them to identify interference problems at much earlier stages of automotive development. Consider, for example, their experience with “decking”—a process where the powertrain and its related components (e.g., exhaust, suspension) are assembled into the upper body of an automobile for the first time (Figure 4). When Chrysler developed the 1993 Concorde and Dodge Intrepid models, decking took more than 3 weeks and required many attempts before the powertrain could be inserted successfully. In contrast, the early use of digital mock-ups in their 1998 Concorde/Intrepid models allowed them to simulate decking and to identify (and solve) numerous interference problems before physical decking took place for the first time. Instead of more than 3 weeks in the 1993 models, Chrysler now could successfully complete the physical decking process in 15 minutes, because all decking problems had already been solved. Thus, by combining new design technology with a different development process, Boeing and Chrysler were able to front-load problem-solving to phases where problems could be identified and solved at a much lower cost and time.

#### *Early Identification of Problems of Function Through Advanced Simulation*

Consider the rapid proliferation of computer simulation in product development and its impact on the early and rapid identification and solving of functional



**Figure 4.** Illustration of “virtual” decking where interference problems are identified and solved with the aid of digital mock-ups before physical assembly takes place.

problems. Until relatively recently, “virtual” experimentation was limited to what could be done using thought experiments and/or calculations that could be done essentially by hand. Experiments that were difficult or impossible to execute by these means were performed using some sort of physical apparatus. However, the rapid improvement of general purpose computers has made it economically possible and desirable to carry out more and more problem-solving via computer simulation.

The advantages of substituting computer models (i.e., simulation) for real physical objects can be very significant. For example, testing automobile structures via real car crashes is expensive and time consuming. In contrast, once set up, a virtual car crash can be run again and again under varying conditions at very little additional cost per run. Further, consider that a real car crash happens very quickly—so quickly that an engineer’s ability to observe details is typically impaired, even given high-speed cameras and well-instrumented cars and crash dummies. In contrast, one can instruct a computer to enact a virtual car crash as slowly as one likes and can zoom in on any structural element of the car (or minute section of a structural element) that is of interest and observe the forces acting on it and its response to those forces during the simulated crash. Thus, if computer simulation is accurate—and there are several areas where its model fidelity is still evolving—it may not only decrease the cost and time of an experimental cycle but also can increase depth and quality of the analysis.

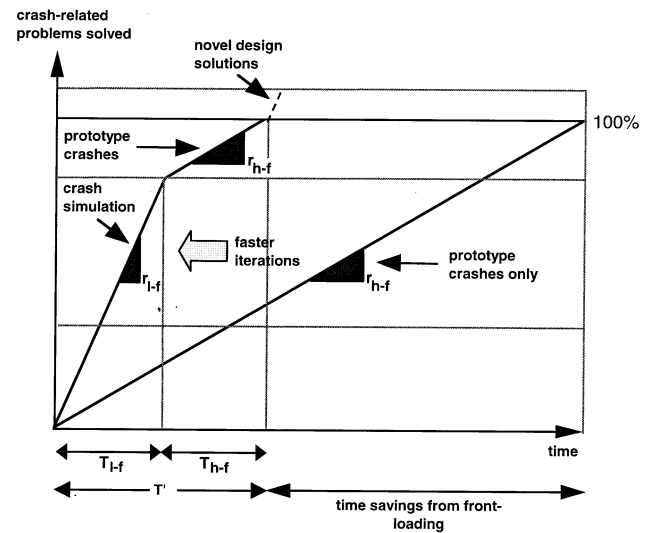
In recent research on automotive development, Thomke [38] found that computer simulation is already having an important impact in the design for crashworthiness and is rapidly changing other important areas as well. His research showed that by speeding up and simultaneously reducing the cost of design iterations, developers were able to increase the frequency of problem-solving cycles while reducing the total time and money spent on the development process. Close examination of an advanced development project (at BMW in Germany) that involved 91 design iterations via crash simulation showed that developers were able to improve side-impact crashworthiness by about 30%—an accomplishment that would have been unlikely with a few lengthy problem-solving cycles using physical prototypes only.

Because of the importance of automotive crashworthiness, the improvements from crash simulation were verified with two physical prototypes that were built after the 91 iterations were completed. Interestingly,

building, crashing, and analyzing the two prototypes cost hundreds of thousands of dollars (as opposed to a few thousand dollars for a simulated crash) and took longer than the entire advanced development project that was studied (Table 1).

Of course, not all crash safety tests can be conducted via simulation because of the complex dynamics necessary to construct very accurate models that would identify all functional problems (examples of tests that are very difficult to model are rollover crashes or crash-related fire hazards). However, identifying and solving a significant percentage of all crash-related design problems more quickly with simulation and then switching to prototype testing later can be a very economical strategy for reasons than we described earlier and is shown in Figure 5.

Thus, being able to test for functionality with the aid of computer simulation is having an important impact on automotive development processes. CAE now affects most areas of product functionality, including acoustics, vibration, aerodynamics, and thermodynamics, and currently is being applied to complex metal-forming processes such as stamping. Developers can test for functionality very early in the process—as early as styling—and receive feedback on functional problems before significant development commitments are made. For example, it is well known that exterior styling and aerodynamic flows are highly



**Figure 5. Combining prototype crashes with simulation can result in faster development and discoveries of novel design solutions.**

interdependent, particularly at high velocities. With computational fluid dynamic (CFD) simulations, stylists now can have their concepts evaluated for aerodynamic problems in a matter of days and can make the necessary changes in a matter of hours.

Finally, we should point out that faster and less costly problem-solving via simulation can open new possibilities for learning and design innovation. The

**Table 1. Approximate Lead Times and Cost for Design Iterations Using Computer Simulation and/or Physical Prototypes in an Advanced Development Project at BMW AG, Germany**

Problem-Solving Step	Simulation <sup>a</sup> (per iteration)	Physical Prototype (per iteration)
(1) Design	<i>Technical Meeting</i> ● <0.5 days	<i>Planning and Piece Part Design</i> ● >2 weeks (involves many meetings)
(2) Build	<i>Data Preparation and Meshing</i> ● Small change: <0.5 days ● Significant change: 1 week ● Entire automobile: 6 weeks	<i>Design and Construction</i> ● Using existing model: 3 months (at \$150,000 per prototype) <sup>b</sup> ● New model: >6 months (at \$600,000 per prototype)
(3) Run	<i>Crash Simulation</i> ● 1 day (varies with computer hardware) @ \$250/day	<i>Crash Physical Prototype</i> ● 1 week (includes preparation of test area)
(4) Analyze	<i>Post-Processing and Analysis</i> ● <0.5 days	<i>Data Preparation and Analysis</i> ● 1 day (crash sensor data only) ● 1-3 weeks (data, crash films, and analysis of physical parts)
Total time	About 2.5 days to 6.3 weeks	About 3.8 months to >7 months
Typical cost (including effort)	About <\$5,000	About >\$300,000

<sup>a</sup> Prior to starting simulation, there is a one-time fixed investment necessary to build a basic model that is used during simulation. This one-time investment is approximately \$100,000 and 2 months, but is not incurred during design iterations.

<sup>b</sup> Prototype build cost and time are a function of the magnitude and number of modifications, but even modest changes can drive up cost and time very quickly. From reference [38].

30% improvement in crashworthiness reported earlier was the direct result of technical innovations that resulted from problems identified with the aid of simulation. Because of the economies of running and the difficulties of analyzing many physical crashes with traditional methods, it would have been very unlikely that some of these novel discoveries had been identified during development. Consider the following example [38] in a study of a design team's problem-solving and learning process:

"In the analysis of prototype crashes of earlier development projects, test engineers repeatedly found that a small section next to the bottom of the center B-pillar 'folded' after a side-impact crash. Extensive testing experience suggested that such 'folding' can result in increased crash barrier penetration and, as a result, in a higher degree of passenger injury. Based on the knowledge and understanding of the underlying crash dynamics, it was commonly assumed that adding metal would strengthen the 'folded' area and thus provide a higher resistance to a penetrating crash object. As prototype crashes had been costly and difficult to evaluate, engineers felt that it was neither necessary nor cost-effective to verify that assumption. However, since simulation was quick, inexpensive and easy to evaluate, one development team member insisted on a verification test. The entire team was very surprised to find out that strengthening the 'folded' area decreased crashworthiness significantly and initially none of the team members had a plausible explanation. However, after careful analysis of the crash data and a detailed study of the underlying crash physics, they learned that an unanticipated secondary and negative effect—the interaction between the 'folded' area and the B-pillar—in fact dominated the primary positive effect that they had anticipated. Equipped with this new knowledge, the team conducted a critical reevaluation of all other enforced areas in the automotive body which led to the improvement of the design for all automobiles currently under development." [38], pp. 67–68]

This suggests that simulation, combined with traditional technologies such as physical prototypes, has the potential to move learning and product performance beyond current levels while cutting both development cost and time.

### **Field Study: Front-Loading Problem-Solving at Toyota**

In the early 1990s, Toyota Motor Corporation, like most other automotive firms, intensified efforts to accelerate their development process. Their objective

was to reduce the time between styling approval and production start, thereby increasing the likelihood that a chosen auto concept would fit with rapidly changing market needs. Among many initiatives such as increased involvement of production during the product concept stage and increased knowledge transfer between projects, Toyota decided to target some development areas where they would make significant investments in CAD/CAE and rapid prototyping capabilities as a means of identifying and solving design problems earlier in their development process. The impact of these initiatives was measured not only in cost and time savings, but more importantly by measuring the changes in problem-solving patterns over time. The data presented in this article are our estimates of Toyota's problem-solving patterns and are based on material shown by Toyota in a public forum [19] and follow-up interviews with Toyota managers in Japan. Because of the strategic importance of front-loading to Toyota, no additional data were available to us at the time the research was conducted.

The front-loading initiatives<sup>7</sup> aimed at identifying and solving design-related problems earlier in the process following styling approval. In their efforts to manage front-loading, Toyota used both of the approaches we suggested earlier: (1) more effective transfer of problem- and solution-specific knowledge between development projects; and (2) early use of advanced technologies such as CAD and CAE. To measure the initiatives' impact, data on problems were collected and tracked over multiple development projects, but each time the intensity of front-loading was increased—more project-to-project knowledge transfer, CAD, and CAE, and even earlier involvement of production engineering in design decisions (Figure 6).

The internal study focused primarily on problems that would be particularly costly and time consuming to solve as they were close to production start and would involve changes such as the modification of production tools. The findings were consistent with expectations: problems were, in fact, identified and solved much earlier in the development process. More specifically, the following observations were made during each round of increasing front-loading activities:

---

<sup>7</sup>Because the term "front-loading" was unknown to them at the time the initiatives were started, Toyota—in the absence of a more suitable term—referred to their internal front-loading activities as simultaneous engineering.

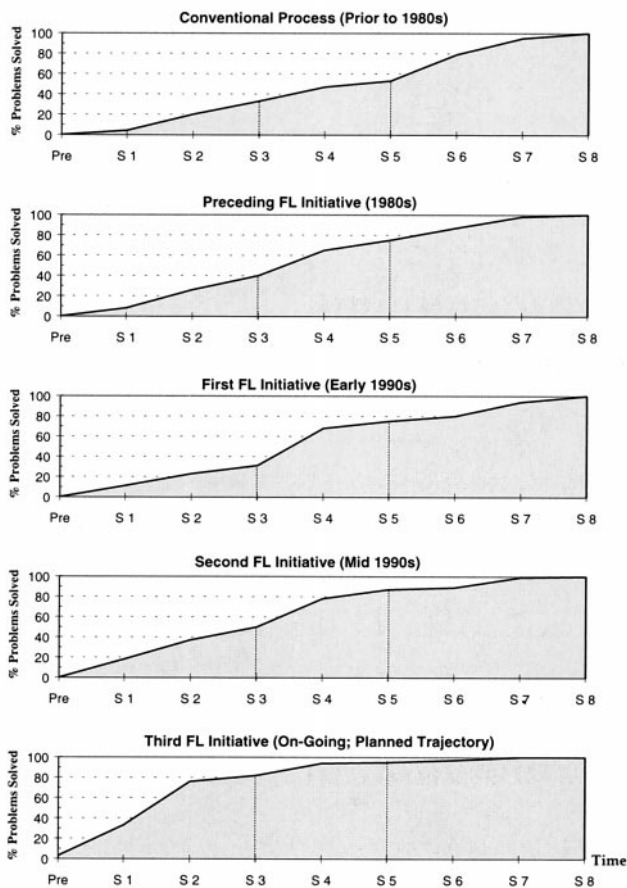


Figure 6. Problem-solving trajectories as a function of development stages and front-loading (FL) intensity (increasing from “conventional” to “third FL initiative”).

- *Conventional process (prior to 1980s)*: As many as a half of the problems remained unsolved at the S5 point, when pilot runs start.
- *Preceding front-loading activities (1980s)*: The communication between prototype shops and production engineering did exist, but it was rather informal and unsystematic. Simultaneous engineering between body engineering and die shops, as well as engines and suspensions, increased, but simultaneous engineering activities among other engineering sections was still lagging behind. As a result, the simultaneous engineering between body and die tended to result in localized problem-solving on body design, but no improvements were made on other components (such as gas tanks, wire harness and connectors, exhaust pipes, clips and bolts).
- *First front-loading initiatives (early 1990s)*: Formal and systematic efforts to improve face-to-face communication and joint problem-solving between pro-

tototype shops and production engineers were made (e.g., formal visits to prototype shops by production engineers). This resulted in an increase of problems identified and solved between the S3 and S4 stages of the development process (Figure 6), increasing the relative percentage of problems found with the aid of first prototypes. Communication between different engineering sections (e.g., between body, engine, and electrical) also were improved.

- *Second front-loading initiative (mid 1990s)*: Move to three-dimensional CAD to identify interference problems (“problems of fit”). As shown in Figure 6, this resulted in an increase of problem identification and solving prior to development stage S3 (first prototypes).
- *Third front-loading initiative (still on-going)*: Move to CAE to identify functional problems earlier in the development process. The transfer of problem and solution information from previous projects to the front end of new projects plays an important role in this initiative as well. As a result, Toyota expects to solve about 80% of all problems by stage S2 or prior to the first prototypes. The second-generation engineering prototypes (S5) only makes a small contribution to solving the total number of problems and thus may be partly eliminated to reduce lead time.

To verify that early problem identification and solving did lead to efficiency and lead time improvements, Toyota examined development cost, number of full physical prototypes built, and lead time over the same periods during which the front-loading initiatives were launched. They found the improvements to be remarkable: Time, cost, and prototype reductions between the first and third front-loading initiatives were reported by Toyota to be in the vicinity of 30–40%. We should note, however, that Toyota also underwent a major reorganization of its development activities in the early 1990s, which resulted in more effective communication and coordination between different areas [12].

Although the data presented did not undergo the same rigor as if we had collected all of it ourselves and thus do not permit us to examine the causal relationship between specific front-loading activities and performance, the data nonetheless present interesting field support for front-loading as an important methodology to improving development performance. According to Toyota managers, many of the recent development process improvements were due to project-to-project knowledge transfer (aided by the increased

use of product platforms) and the effective and early use of computer and rapid prototyping technologies. Toyota managers are now applying the lessons learned from these initiatives to other development areas, with the objective of identifying many potential engineering and production problems as early as the concept phase.

**Discussion**

This article introduces the concept of front-loading problem-solving, which we define as a strategy that seeks to increase development performance by shifting the identification and solving of [design] problems to earlier phases of a product development process. We started our discussion by describing product development as a bundle of problem-solving cycles. Using this view, we proposed that faster product development can be achieved with an earlier generation of problem- and solution-related information, particularly if it involves critical path activities. Such information may be obtained from previous projects or other repositories of developmental knowledge if it already exists, or with new problem-solving cycles if the information has to be generated. With the aid of a conceptual framework and field observations, we examined two approaches to shifting problem-solving trajectories upstream:

- *Project-to-project knowledge transfer*: increase the initial number of problems solved (or avoided) by more effective project-to-project transfer of problem and solution-specific information;
- *Rapid problem-solving*: leverage advanced technologies (such as CAD and CAE) or other methods to increase the overall rate of development problems identified and solved.

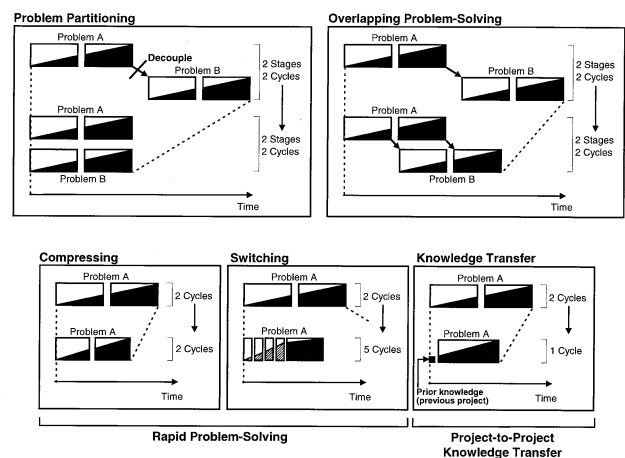
These approaches were described in detail and supported with examples and empirical observations from development practice.

*Other Approaches to Front-Loading Problem-Solving*

Although we presented two approaches to front-loading, it is important to note that other approaches can also lead to upstream shifts in problem-solving. Complementary methodologies already reported in the literature and mentioned earlier in the article include the overlapping and optimal partitioning of development tasks to reduce development lead time. For example,

*rapid problem-solving* focuses on the more rapid and efficient completion of a given development task through faster problem-solving cycles. In contrast, the overlapping or concurrent completion of individual tasks can lead to early problem-solving by starting downstream problem-solving earlier than in a sequential development process (see Figure 7 for a comparison of different approaches).

As suggested by Figure 7, a problem-solving perspective of product development can help in understanding how different approaches to improving development performance complement each other. Similarly, seemingly unrelated approaches from other research can be examined in terms of their impact on the upstream shift of problem-solving activities. A recent research example illustrates how a well-known managerial problem—the structure of explicit and implicit contracts and reward systems—can be examined in the context of changes in problem-solving patterns. A study by Garel and Midler [15] on contractual relationships between European automotive firms and their suppliers showed that contracts can affect a firm’s problem-solving patterns. Under traditional contracting, suppliers could charge any design changes due to problems discovered in the late tool production phase to the automotive customer, irrespective of who originated the change. According to Garel and Midler, these late and costly tool changes generated an average of 20% in additional revenues for suppliers. Not surprisingly, the same suppliers had



**Note:** Each rectangle stands for one problem-solving cycle. The horizontal axis measures the percentage of total problems solved. For simplicity, it is assumed that (1) the number of problems solved increases linearly, and (2) only two modes of problem-solving exist (simulation and physical prototyping).

Physical Prototype  
Simulation ("Virtual")

**Figure 7. Summary of complementary approaches that can lead to an upstream shift of problem-solving and, as a result, increased development performance.**

little incentive to co-operate with their customer or invest in technologies or methods that would result in early problem-solving. The researchers then compared these traditional contracts with a recent project where the contract had provisions that would penalize suppliers for tool changes late in the design process. As a result, the suppliers became very active in exchanging problem and solution-specific information with the customer as early as possible and, as the results showed, focused on decreasing late and costly tooling changes. In their comparison, Garel and Midler found a remarkable cost difference: not only was the new contract more cost effective for both parties involved but also resulted in significant differences in the timing of problem-solving. Under the traditional contract, 49% of estimated tooling costs were due to post-design freeze tooling changes, compared to only 15% under the revised contract. This suggests that other approaches, such as the change in contracting described, can lead to important changes in problem-solving patterns for firms and thus provide opportunities for further research undertakings.

#### *Role of Front-Loading Problem-Solving in Building Development Capabilities*

It is important to understand the role that front-loading problem-solving can play in building organizational capabilities for increased development performance, or as Nelson [27] states with respect to core capabilities: “At any time the practiced routines that are built into an organization define a set of things the organization is *capable* of doing confidently. If the lower-order routines for doing various tasks are absent, or if they exist but there is no practiced higher-order routine for invoking them in the particular combination needed to accomplish a particular job, then the capability to do that job lies outside the organization’s extant *core capabilities*.” But how can a firm measure the presence and/or any improvements of such lower- and higher-order routines? Certainly, one possibility is a focus on indirect measures such as time, resources, and product quality. Organizations with superior development capabilities should do well along all of these dimensions. But like a plant manager who may have difficulty in improving the hundreds or more manufacturing routines by focusing on aggregated plant level variables only, development managers can benefit from more direct measures of changes in capabilities. We propose that the systematic measurement of problem-solving, as described in the Toyota

data, can provide a more direct focus on capability building.

As an example, consider that many automotive firms currently are trying to reduce development times by aggressively substituting physical prototypes with computer models. (An example would be the elimination of second-generation hardware prototypes.) However, without an explicit understanding of their problem-solving capabilities, firms do not know if they have all the lower- and higher-order routines in place that would allow them to reduce time without sacrifices in product quality; as measured by the number of unidentified problems after releasing a product to market. In contrast, the management of problem-solving trajectories (as shown in Figure 6) that we propose in this article may help firms to measure more directly changes in their capabilities over time, using some of the approaches that we described. (Using the automotive example once more, second-generation hardware prototypes ought to be eliminated after—and not before—changes in development capabilities have been measured and validated through changes in problem-solving trajectories.) Shifting problem-identification and problem-solving to earlier phases in development thus can be an explicit objective, whereas faster development will be an indirect benefit of having solved more problems at these earlier stages.

---

We would like to thank the many colleagues who contributed to the ideas expressed in this article through private conversations, seminars, and feedback on earlier drafts of this paper. The financial support of the Harvard Business School Division of Research and Tokyo University’s Division of Economics are gratefully acknowledged.

---

#### References

1. Adler, P. Interdepartmental interdependence and coordination: The case of the design/manufacturing interface. *Organization Science* 6 (1995).
2. Aoshima, Y. Knowledge Transfer Across Generations: The Impact on Product Development Performance in the Automobile Industry. Unpublished Ph.D. Dissertation. Massachusetts Institute of Technology, 1996.
3. Argyris, C. Reasoning, Learning, and Action. San Francisco: Jossey-Bass Publishers, 1982.
4. Allen, T. J. Studies of the problem-solving process in engineering design. *IEEE Transactions on Engineering Management* EM-13:72–83 (1966).
5. Boehm, B. Software Engineering Economics. Englewood Cliffs: Prentice Hall, 1981.
6. Brooks, F. P. The Mythical Man-Month. Reading, MA: Addison-Wesley, 1982.

7. Brown, S. and Eisenhardt, K. Product development: Past research, present findings, and future directions. *Academy of Management Review* 20:343–378 (1995).
8. Clark, K. and Fujimoto, T. Lead time in automobile development: Explaining the Japanese advantage. *Journal of Technology and Engineering Management* 6:25–58 (1989).
9. Clark, K. and Fujimoto, T. *Product Development Performance*. Boston: Harvard Business School Press, 1991.
10. Cooper, Robert and Kleinschmidt, E. Determinants of timeliness in product development. *Journal of Product Innovation Management* 11:381–396 (1994).
11. Crawford, M. The hidden cost of accelerated product development. *Journal of Product Innovation Management* 9:188–199 (1992).
12. Cusumano, M. and Nobeoka, K. *Thinking Beyond Lean: How Multi-Project Management is Transforming Product Development at Toyota and Other Companies*. New York: The Free Press, 1998.
13. Cusumano, M. and Selby, R. *Microsoft Secrets*. New York: The Free Press, 1995.
14. Fujimoto, T. *Organizing for Effective Product-Development: The Case of the Global Automobile Industry*. Unpublished Dissertation. Harvard Business School, 1989.
15. Gareil, G. and Midler, C. An analysis of co-development performance in automotive development processes: A case study testing a win-win hypothesis. 5th EIASM International Product Development Conference, Como, Italy, 1998.
16. Griffin, Abbie. Modeling and measuring product development cycle time across industries. *Journal of Engineering and Technology Management* 14:1–24 (1997).
17. Gupta, A. and Wilemon, D. Accelerating the development of technology-based new products. *California Management Review* 32:24–44 (1990).
18. Iansiti, M. *Technology Integration: Making Critical Choices in a Turbulent World*. Boston: Harvard Business School Press, 1997.
19. Jagawa, T. Frontloading: Shortening development time at Toyota through intensive upfront effort. IBEC 95, Detroit, Michigan, 1995.
20. Kamien, M. I. and Schwartz, N. L. *Market Structure and Innovation*. Cambridge: Cambridge University Press, 1982.
21. Krishnan, V., Eppinger, S., and Whitney, D. A model-based framework to overlap product development activities. *Management Science* 43 (1997).
22. Liker, Jeffrey, Fleischer, Mitchell, Nagamachi, Mitsuo, and Zonneville, Michael. Designers and their machines: CAD use and support in the US and Japan. *Communications of the ACM* 35 (1992).
23. Marples, D. L. The decision of engineering design. *IRE Transactions on Engineering Management* 2:55–71 (1961).
24. McDonough, Edward, III, and Barczak, Gloria. The effects of cognitive problem-solving orientation and technological familiarity on faster new product development. *Journal of Product Innovation Management* 9:44–52 (1992).
25. Meyer, C. *Fast Cycle Time*. New York: The Free Press, 1993.
26. Meyer, M. H. and Lehnerd, A. *The Power of Product Platforms*. New York: The Free Press, 1997.
27. Nelson, R. Why do firms differ, and how does it matter. In: *Fundamental Issues in Strategies*. R. Rummelt, D. Schendel, and D. Teece (eds.). Boston: Harvard Business School Press, 1994.
28. Petroski, H. *To Engineer Is Human*. New York: Vintage Books, 1992.
29. Pisano, G. *The Development Factory*. Boston: Harvard Business School Press, 1996.
30. Sabbagh, K. *Twenty-First Century Jet: The Making and Marketing of the Boeing 777*. New York: Scribner, 1996.
31. Scott-Morton, M. *Computer-Driven Visual Display Devices*. Unpublished Doctoral Dissertation. Harvard Business School, 1967.
32. Senge, P. *The Fifth Discipline: The Art and Practice of the Learning Organization*. New York: Doubleday, 1990.
33. Simon, H. A. *The Sciences of the Artificial*. Cambridge: MIT Press, 1969.
34. Smith, P. and Reinertsen, D. *Developing Products in Half the Time: New Rules, New Tools*. New York: Van Nostrand Reinhold, 1998.
35. Terwiesch, C. and Loch, C. Measuring the effectiveness of overlapping development activities. Working Paper No. 98/45/TM, INSEAD, Management Science (1998).
36. Thomke, S. Managing experimentation in the design of new products. *Management Science* 44:743–762 (1998).
37. Thomke, S. The role of flexibility in the development of new products: an empirical study. *Research Policy* 26:105–119 (1997).
38. Thomke, S. Simulation, learning and R&D performance: Evidence from automotive development. *Research Policy* 27:55–74 (1998).
39. Thomke, S. and Bell, D. Optimal testing in product development. Harvard Business School Working Paper No. 99-053 (1998).
40. Thomke, S. and Nimgade, A. BMW AG: The Digital Auto Project (A). Harvard Business School Case Study No. 699-044 (1998).
41. Verganti, Robert. Leveraging on systemic learning to manage the early phase of product innovation projects. *R&D Management* (1997).
42. von Hippel, E. “Sticky” information and the locus of problem-solving: Implications for innovation. *Management Science* 40:429–439 (1994).
43. von Hippel, E. and Tyre, M. How “learning by doing” is done: Problem identification in novel process equipment. *Research Policy* 19:1–12 (1994).
44. von Hippel, E. Task partitioning: An innovation process variable. *Research Policy* 19:407–418 (1990).
45. Watkins, M. and Clark, K. Strategies for managing a project portfolio. Working Paper, Harvard Business School (1994).
46. Wheelwright, S. and Clark, K. *Revolutionizing Product Development*. New York: The Free Press, 1992.