CIRJE-F-11

Shortening Product Development Time through "Front-Loading" Problem-Solving

Stefan Thomke
Harvard University

Takahiro Fujimoto
The University of Tokyo

July 1998

SHORTENING PRODUCT DEVELOPMENT TIME THROUGH "FRONT-LOADING" PROBLEM-SOLVING

Stefan Thomke[1] and Takahiro Fujimoto[2]

[1]*Assistant Professor*
Harvard University, Graduate School of Business Administration, Boston, MA 02163
Tel. +1-617-495-6569, Fax +1-617- 496-4072
sthomke@hbs.edu

[2]*Associate Professor*
University of Tokyo, Faculty of Economics, Tokyo 113
Tel. + 81-3-3812-2111, Fax +81-3-3818-7082
fujimoto@e.u-tokyo.ac.jp

ABSTRACT

Shorter development lead times, other things being equal, contribute to competitive advantages in many of today's industries where market needs and available technologies are difficult to predict, rapidly changing or diversified and where novelty is highly valued in the market. In this paper, we focus on the identification and solving of development problems during earlier phases of product development - a concept that we define as *front-loading* - as a way to reduce development time and cost and thus free up resources to be more innovative in the marketplace. Using a problem-solving perspective of product development, we develop a conceptual model of front-loading and present related examples and case evidence from development practice. We propose that front-loading can be achieved using multiple approaches out of which the following two are presented in detail: (1) increased problem and solution-specific knowledge transfer between projects; and (2) higher rates of early problem-identification and solving through the effective deployment of advanced technologies and methods. The implications for managerial practice and theory are discussed and suggestions for further research are proposed.

"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

Shortening product development time has become an explicit objective of many firms, particularly in very competitive industries where time has become a key driver for development success. The importance of shorter development time has been supported by a number of publications, ranging from the strategic value of time [27] to methodologies and techniques that can aid managers in achieving significant time reductions ([26], [37]). Some of the proposed methods include adding people [5], finding and focusing on critical paths, overlapping problem-solving ([6], [15], [28]), and the utilization of carry-over parts and platforms ([7], [17], [18]).

In this paper we have adopted a problem-solving perspective which is increasingly recognized by researchers as being fundamental to product development. Managers of development projects are usually very concerned about the identification of problems since 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 as to move (or "**load**") their problem identification and solution backwards in time (to the "**front**" of the process). In this paper, we will provide a conceptual model and some evidence from development practice to describe the basic principle behind this rapidly emerging concept.

We base our discussion on a problem solving perspective of product development. We will describe and analyze a product development project as a bundle of inter-dependent problem-solving cycles that include modeling and testing via computers or physical prototypes as core activities. Using this perspective, we propose that the *earlier identification and solving of a set of problems for a given developmental task* can lead to faster and more efficient product development. While we present examples of front-loading from different industries, we will

primarily report on findings from the automotive sector where the methodology is currently being applied to the reengineering and shortening of product development. Our evidence is mostly case-based but we feel it points towards the emergence of an important capability for improving development practice.

The paper will proceed as follows. We start with a brief discussion of the benefits and potential pitfalls of shortening product development. We then describe the problem-solving perspective of product development and apply it to automotive development. The next sections explain the basic principle of 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 paper with implications for managerial practice and research.

COMPETITIVE BENEFITS OF SHORTENING DEVELOPMENT TIME

Shorter development lead time, other things being equal, contributes to competitive advantages in many of today's industries where market needs and available technologies are difficult to predict, rapidly changing or diversified, where novelty is highly valued in the market, and where competition through new products and technologies are intense. There are both direct and indirect effects resulting from faster development. Shorter lead times may *directly* lead to successful product development because: (i) earlier start of product shipment brings about additional profits in the situation of high market growth and short product lives due to rapid changes in both needs and technologies [26][1]; (ii) the first introducer of a new product enjoys a high monopoly rent when consumers are willing to pay high prices for its novelty [14]; (iii) shorter lead times enable planners to create product concepts with more accurate forecasts of future market needs ([7],[37]). (Shorter intervals between concept fix and market introduction tend to make the market need forecast more accurate, and thereby make the product concept fit needs better, other things being equal.)

Shorter lead times also have profound *indirect* effects through the enhancement of organizational capabilities. *First*, while managers often assume a trade-off between lead times

---

[1] In a frequently cited analysis, Reinertsen [21] demonstrated the importance of development speed. Using a very simple economic model of new-product development and commercialization, he found that in high- and low-growth markets, overrunning development cost by 50% had less than a 4% impact on profits before tax. Assuming a high growth market with short product life cycles provided some very surprising results: shipping a product six months late can draw down its life cycle profits

and development costs, some recent empirical research found just the opposite to be true. In a world-wide study of automotive development, researchers found that the same set of organizational capabilities that was responsible for shorter lead times also had an impact on development efficiency. Firms that developed automobiles faster also tended to be more efficient at it [7] (see **figure 1**). Thus, faster development can free up resources which, in turn, can be used to develop better products. *Second,* fast product development also enables more effective organizational learning because feedback on managerial actions is fast and less ambiguous [24] and it enables firms to adapt more rapidly to best practice or changes in the environment.

INSERT FIGURE 1 ABOUT HERE

Of course, there also risks that result from an emphasis on development speed. Firms that choose to shorten product-life cycles by more frequent new product introductions may observe a negative impact on future cumulative revenue [35]. Or too much organizational focus on time reductions may have undesirable consequences for product quality. This can happen in a number of ways. By pushing very hard for time reductions without developing the necessary process and organizational capabilities that would allow such reductions, managers will force developers to make trade-offs which may result in lower product quality. The purpose of this paper is to show how such capabilities can be developed gradually such that time and cost reductions can be achieved without sacrifices in product quality. In fact, companies successful at front-loading will find that some of the methods discussed in this paper can help them to develop better products with more innovative content.

A PROBLEM-SOLVING PERSPECTIVE OF PRODUCT DEVELOPMENT

While the early innovation and R&D management literature views problem-solving as a fundamental design activity ([3], [16], [25]), it has only been in recent years that problem-solving is being discussed and explicitly adopted in the product development literature (for example, see [6], [29], [31],[37]). In this paper, we view problem-solving as an

---

by 33%. This finding did not hold in a slow-growth market with long product-life cycles: late shipment created only a 7% decline in these profits.

iterative process, driven by trial and error experiments that are guided by knowledge of underlying relationships between cause and effect (see **figure 2**).

INSERT FIGURE 2 ABOUT HERE

Problem-solving starts with problem recognition and goal definition and continues with an iterative process of experimental search through alternatives that are designed and build during **step 1** (design) and **step 2** (build models) of a four step problem-solving cycle[2]. These alternatives may or may not include the best possible solutions - one has no way of knowing. The alternatives are then 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 towards 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, the experimental conditions, the fidelity of the experimental set-up 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" ([2], [19]).

Applying the problem-solving perspective to automotive development (**figure 3**), one finds that it consists of a bundle of numerous *problem solving cycles*, each of which consists of design, build, test 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,

---

[2] Similar building blocks to analyze the design and development process were used in earlier research. Simon [25] examined design as series of "generator-test cycles". Clark and Fujimoto [6] and Wheelwright and Clark ([37]; chapters 9 & 10) used "design-build-test" cycles as a framework for problem-solving in product development. Thomke [29] modified the blocks to include "run" and

involving several development groups (such as major prototyping cycles). As projects progress, cycles tend to include models of increasing fidelity (e.g., thought experiments, computer simulations, physical prototypes, pilot vehicles, etc.) for testing the effect of design decisions on functionality, geometric fit and manufacturability [10]. Problem solving cycles can be structured in hierarchical form: they are iterated to complete a *task* that creates a solution for each component; tasks are then integrated into major *stages* of development such as product engineering and process engineering.

INSERT FIGURE 3 ABOUT HERE

EARLY PROBLEM-SOLVING THROUGH "FRONT-LOADING"

An important part of an effective development strategy is the timing and fidelity of test models (e.g. simulation models, prototypes, etc.) [37] . 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 provides 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 [4]. 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, revalidation. 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. While considerable progress to make development processes more flexible has been made in recent years ([12], [30]), late engineering changes as a response to identified design problems can

"analyze" as two explicit steps that conceptually separate the execution of an experiment and the learning that takes place during analysis.

still be very costly and time-consuming. This is particularly true in automotive development where late design changes can cost millions of dollars and can take weeks or months to be carried out - partially due to increasing tooling commitments.

What is Front-Loading?

We define front-loading as *a strategy that seeks to reduce development time and cost by shifting the identification and solving of [design] problems to earlier phases of product development.* We propose that effective front-loading can be achieved using multiple approaches out of which two will be presented in detail. The *first* approach involves the effective transfer of problem- and solution-specific information between development projects as to reduce the total number of problems 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 involves the effective deployment of advanced technologies and methods which directly influence the overall rate at which 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 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 more problem-solving cycles over a shorter period of time.

Front-Loading Problem-Solving: A Taxonomy and a Conceptual Model

To solve a problem, problem and solution-related information has to be created, made available and recognized by the problem-solver. Studies of the problem identification process have shown 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 [34].

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 is generally available to them but they are unable to predict the often subtle chain of cause and effect that lead to a design problem. Indeed, in their study of 27 field problems that affected two novel process machines, von Hippel and Tyre [34] found that 15 out 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, and in the remaining 5 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 of course desirable to create such information as early as possible. One could, for example, shift ample resources such as more personnel and increased testing budgets to the very early stages of a new development project. However, development practice usually looks different: resources are ramped up slowly as a project unfolds and thus problem-solving activities and the related creation of information ramps up slowly as well. Hence, patterns of problem identification are shifted downstream ("end-loaded"). The objective of this paper is to present an approach - front-loading - that reverses this effect and shifts problem-identification and solving upstream.

To conceptualize front-loading with the aid of a simple model, consider a product development process where problems are identified and solved in an iterative fashion, following the design-build-run-analyze cycles described earlier (**figure 4**). Suppose that we develop in an environment where only one kind of high-fidelity prototype can be build and no problem or solution-specific information is being transferred between projects (see **figure 4; case (a)**). Problems are identified and solved at a constant rate a_2 that depends on the speed at which a d-b-r-a cycle can be completed[3]. Total development time T can only be shortened if the problem-solving rate a_2 is increased by carrying out d-b-r-a cycles more quickly. Quite often, these cycles can be accelerated 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.

---

[3] To keep the model simple, we are using a first-order approximation. Thomke [29] showed that the problem-identification rate is more likely to experience diminishing returns and thus follow a non-linear trajectory.

However, if a second class of prototypes of lower fidelity but a higher problem-solving rate a_1 was available and knowledge is transferred between projects (**figure 4; case (b)**), developers could use the following two approaches to shift the problem-solving trajectory:

- increase the initial number of problems solved (m_0) by more effective project-to-project transfer of problem and solution-specific information.
- use the low-fidelity prototype to solve problems that it can identify (m_1) and then switch to the higher fidelity prototype for the remaining problems (m_2).

INSERT FIGURE 4 ABOUT HERE

The combined benefit will be a shorter total development time T as shown in figure 4, case (b). In general, management should extend the same logic to determine an optimal number of prototypes (n), each with a different fidelity, such that total development time is minimized. The price one would pay for a large n is the cost of developing and building many different kinds of prototypes and thus foregoing the economies of scale and specialization from building one kind of prototype only. The opportunities that advanced technologies such as computer simulation, three-dimensional CAD and rapid prototyping can provide now become quite apparent: even though they sometimes provide lower fidelity than physical prototypes, they can identify a sizable percentage of problems at a rate significantly higher than conventional high-fidelity prototypes. Combined with conventional prototyping, we suggest that these technologies can result in remarkable development time reductions through earlier problem solving.

MANAGING FRONT-LOADING: EVIDENCE FROM INDUSTRIAL PRACTICE

In the previous section, we have identified two general approaches to the management of front-loading activities that would result in a shift of problem-identification and solving to earlier phases of product development. The *first* approach involved the transfer of problem- and solution-specific knowledge between development projects as to reduce the total number of problems to be solved from the onset of development activities. The *second* approach addressed the effective deployment of advanced technologies or methods which directly influence the rate at which problems are identified and solved. As

figure 4 shows, both of these approaches can result in an upstream shift of problem-solving and thus enable firms to reduce development time and cost significantly. We will now present examples from development practice that demonstrate the benefits of both approaches.

Project-to-Project Transfer of Problem and Solution-Specific Knowledge

Studies of problem-solving have shown that firms often find "old" problems in new development projects. In their study of the development of front and rear body closures (i.e. hoods, trunk-lids, and liftgates), Watkins and Clark [36] found that design problems were often 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 [34] observed a similar pattern and found that not only was there a lack of problem-specific information being transferred but designers were sometimes unable to use the transferred information effectively. Thus, it appears that a more effective transfer of knowledge between projects can improve development performance [1, 18]. In this paper, we are particularly interested in the transfer of problem and solution-specific information which can reduce the overall number of problems for a new project, as shown in **figure 4**.

As a 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 others, information on problems discovered at various stages of software development. Cusumano and Selby [9] reported that much of the learning between projects at Microsoft can be attributed to its systematic use of such postmortem reports. In their research on Microsoft they found that development teams generally take three to six months to prepare a postmortem which can be between under 10 to over 100 pages long. Besides accounting for people, product and scheduling issues, the postmortems also contain detailed information on number of problems [bugs] identified, problem severity, record of finding and solving problems, and so forth. Preparing, discussion 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 [9]. Equipped with such information, developers can move more quickly

towards the early identification and solution of problems that seemed novel at first but have been 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. That, however, may not always be the case. In a study of 229 project members of 25 Japanese automobile projects, Aoshima [1] found that effective knowledge transfer mechanism 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, etc.). However, if the knowledge is "sticky" [33] or very costly to encode and transfer, such as tacit knowledge about the integration of components, firms performed better if they relied face-to-face communication, people transfers, and other mechanisms that allow for an effective transfer of such tacit knowledge.

Earlier and Faster Problem-Solving Cycles 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 - or in other words, they don't 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. However, as we will see, the technologies and process changes chosen depend on the problems being solved.

Early identification of problems of <u>fit</u> through digital mock-ups

Designers are often 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 often involve thousands or hundred-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 [22][4]. Experience has 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." (Sabbagh [22])

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 computer-aided design (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

---

[4] The following description of Boeing's development practice is based on Sabbagh's [22] detailed account of the 777 aircraft project.

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 which 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 which resulted in 251 interference problems – problems that would have been very costly and time-consuming to correct during final assembly[5].

Other firms are currently experiencing the same benefits from using advanced technology to front-load problem-solving. Chrysler Corporation discovered 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 (see **figure 5**). When Chrysler developed the 1993 Concorde and Dodge Intrepid models, decking took more than three weeks and required many attempts before the powertrain could be successfully inserted. 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 three weeks in the 1993 models, Chrysler could now successfully complete the physical decking process in 15 minutes as 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.

INSERT FIGURE 5 ABOUT HERE

[5] We should also 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 [22]). The unusual interaction arose because interference testing via three-dimensional CAD forced designers to interact and solve integration problems early.

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 identification and solving of functional 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, studying automobile structures via real car crashes is quite 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 experiment happens very quickly - so quickly that the experimenter'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 crash. Thus, if computer simulation is accurate - and there are still several areas where its fidelity is still evolving- it may not only decrease the cost and time of an experimental cycle but can also increase depth and quality of the analysis (step(4)).

In recent research on automotive development, Thomke [31] found that computer simulation is already having a dramatic impact in at least one very critical area - 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 iterations 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 prototype iterations.

Because of the importance of automotive crashworthiness, the improvements from crash simulation were verified with two physical prototypes that were build 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 (see **table 1**).

INSERT TABLE 1 ABOUT HERE

Of course, not all crash tests can be conducted via simulation because of the complex dynamics that are necessary to construct very accurate models that would identify all functional problems. However, identifying and solving a significant percentage of all crash-related design problems more quickly with simulation and then switch to prototype testing later can be a very economical strategy for reasons than we described earlier and is shown in **figure 6**.

INSERT FIGURE 6 ABOUT HERE

Thus being able to test for functionality with the aid of computer simulation is having an important impact on automotive development processes. Computer-aided engineering now includes most areas of product functionality, including acoustics, vibration, aero- and thermodynamics, and is now 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-know that exterior styling and aerodynamic flows are highly interdependent. With computational fluid dynamic (CFD) simulations, stylists can now have their concepts evaluated for aerodynamic problems in a matter of days and 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 design innovation. The 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[6]. This suggests that simulation has the potential to move 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 computer-aided design and engineering (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 paper are our estimates of Toyota's problem-solving patterns and are based on material shown by Toyota in a public forum [13] and follow-up interviews with Toyota managers in Japan. Because of the strategic importance

---

[6] As an example of how the use of simulation can lead to novel discoveries and insights, consider the following example from Thomke [31]:
"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 ."

of front-loading to Toyota, no additional data was available to us at the time the research was conducted.

The front-loading initiatives[7] aimed at identifying and solving design-related problems earlier in the process following styling approval. In their efforts to manage front-loading, Toyota employed both of the approaches the 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 was 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 (**figures 7.a - 7.e.**).

INSERT FIGURE 7 ABOUT HERE

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 (FL) activities ((a) through (e)):

(a) ***Conventional process*** *(prior to 1980s)*: As many as a half of the problems remained unsolved at the S5 point, when pilot runs start.

(b) ***Preceding FL activities*** *(1980s)*: The communication between prototype shops and production engineering did exist, but it was rather informal and unsystematic. Simultaneous engineering (SE) between body engineering and die shops, as well as engines and suspensions, increased but SE 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 (e.g., gas tanks, wire harness and connectors, exhaust pipes, clips and bolts).

---

[7] Since the term "front-loading" was unknown to them at the time the initiatives were started, Toyota referred to their internal front-loading activities as simultaneous engineering (SE).

(c) ***First FL initiatives*** *(early 1990s)*: Formal and systematic efforts to improve face-to-face communication and joint problem solving between prototype 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, 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) were also improved.

(d) ***Second FL initiative*** *(mid 1990s)*: Move to three-dimensional computer-aided design (CAD) as to identify interference problems ("problems of fit"). As figure 7(d) shows, this resulted in a significant increase of problem identification and solving prior to development stage S3 (first prototypes).

(e) ***Third FL initiative*** *(still on-going)*: Move to computer-aided engineering (CAE) as 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 ("knowledge front-loading") 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.

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 quite remarkable: both cost and time were reduced significantly (see **table 2**). We should note that Toyota underwent a major reorganization of its development activities in the early 1990s which resulted in more effective communication and coordination between different areas [8]. Some of these performance improvements observed, particularly during the first front-loading initiatives (early 1990s), were most likely affected by these changes.

INSERT TABLE 2 ABOUT HERE

While the data presented did not undergo the same rigor than if we had collected *all* of it ourselves, it nonetheless presents some very interesting field support for front-loading as an important methodology to improving development performance. According to Toyota

managers, most of the recent performance 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 paper introduces the concept of front-loading which we define as a strategy that seeks to reduce development time and cost by shifting the identification and solving of [design] problems to earlier phases of product development. Recognizing the importance of managing problem-solving in the improvement of development performance, we have focused on the effective use of recent technologies such as advanced computer-aided design and engineering (CAD/CAE), combined with the project-to-project transfer of problem- and solution-specific knowledge.

We started our discussion by describing product development as a bundle of problem-solving cycles. Using this view we proposed that shorter lead times would call for an earlier creation 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 model and field observations, we developed two approaches to shifting problem-solving trajectories upstream:

- increase the initial number of problems solved by more effective project-to-project transfer of problem and solution-specific knowledge;
- use advanced technologies (such as computer-aided design and engineering) or other methods to increase the overall rate of problem-solving.

The approaches were described in detail and supported with examples and empirical data from development practice.

The Role of Front-Loading in Building Capabilities

It is important to understand the role that front-loading can play in building organizational capability for increased development performance. For example, many automotive firms are currently trying to reduce development times by aggressively substituting

physical prototypes with computer models. (An example would be the forced elimination of second generation hardware prototypes.) However, without an explicit understanding of their problem-solving trajectories, firms do not know if they have the organizational capabilities in place that would in fact allow them to reduce time without sacrifices in quality; as measured by the number of unidentified problems before releasing a product to market. In contrast, the management of problem-solving trajectories (as shown in **figure** 7) that we propose in this paper allows firms to gradually build their capabilities over time, using the approaches that we described. (Using the example above once more, second generation hardware prototypes would be eliminated after - and not before - the organizational capability has been validated through changes in problem-solving curves.) Shifting problem-identification and solving to earlier phases in development will be an explicit objective whereas faster development will be an indirect benefit of having solved more problems at these earlier stages.

Our field research supports the importance of organizational capabilities as central to achieving world-class performance. One Japanese company we studied used front-loading techniques to cut their development lead time significantly. They achieved this remarkable change through strong leadership of a senior executive who focused the organization on the management of problem-solving curves. Advanced technology played an important but secondary role as no supercomputers were used in the firm's CAE activities. Instead the company found that integrating new knowledge into its development process and rapidly responding to feedback from problem-solving cycles was key to shifting their problem-solving curve upstream. Considering that the adoption of advanced technologies can often be carried out more quickly than changes to an organization and its processes, we thus propose that competitive advantage derived from technology *alone* may not be sustainable in the long run.

Other Approaches to Front-Loading Problem-Solving

While we have presented two approaches to front-loading, it is important to note that there are other mechanisms that can lead to a shift in problem-solving to earlier phases of development. An important example is the structure of explicit and implicit contracts and reward systems as they play an integral role in how different agents plan and carry out their problem-solving activities. Thus making changes to contracts in order to emphasize early problem-solving can have significant development cost and time consequences. A recent study by Garel and Midler [11] of 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 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 cost were due to post-design freeze tooling changes, compared to only 15% under the revised contract. This suggests that other approaches to front-loading, such as the contracting approach described, can lead to important changes in problem-solving patterns for firms and thus provide opportunities for further research undertakings.

REFERENCES

[1] Aoshima, Y., 1996, Knowledge Transfer Across Generations: The Impact on Product Development Performance in the Automobile Industry (Unpublished Ph.D. Dissertation, Massachusetts Institute of Technology).

[2] Argyris, C., 1982, Reasoning, Learning, and Action (Jossey-Bass Publishers, San Francisco).

[3] Allen, T.J., 1966, Studies of the problem-solving process in engineering design, IEEE Transactions on Engineering Management EM-13, no. 2, 72-83.

[4] Boehm, B., 1981, Software Engineering Economics (Prentice Hall, Englewood Cliffs).

[5] Brooks, F. P., 1982, The Mythical Man-Month (Addison-Wesley, Reading, Massachusetts).

[6] Clark, K. and T. Fujimoto, 1989, Lead time in automobile development: explaining the Japanese advantage, Journal of Technology and Engineering Management, 6, 25-58.

[7] Clark, K. and T. Fujimoto, 1991, Product Development Performance (Harvard Business School Press, Boston).

[8] Cusumano, M. and K. Nobeoka, 1998, Thinking Beyond Lean: How Multi-Project Management is Transforming Product Development at Toyota and Other Companies (forthcoming, Free Press/Simon & Schuster).

[9] Cusumano, M. and R. Selby, 1995, Microsoft Secrets (The Free Press, New York).

[10] Fujimoto, T., 1989, Organizing for Effective Product-Development - The Case of the Global Automobile Industry (Unpublished Dissertation, Harvard Business School).

[11] Garel, G. and C. Midler, 1998, 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.

[12] Iansiti, M., 1997, Technology Integration: Making Critical Choices in a Turbulent World (Harvard Business School Press, Boston).

[13] Jagawa, T., 1995, Frontloading: shortening development time at Toyota through intensive upfront effort, IBEC 95 (Detroit, Michigan).

[14] Kamien, M.I. and Schwartz, N.L., 1982, Market Structure and Innovation (Cambridge University Press, Cambridge).

[15] Krishnan, V., S.Eppinger and D. Whitney, 1997, A model-based framework to overlap product development activities, Management Science, vol. 43 no. 4.

[16] Marples, D.L., 1961, The decision of engineering design, IRE Transactions on Engineering Management 2, June, 55-71.

[17] Meyer, M.H. and A. Lehnerd, 1997, The Power of Product Platforms (The Free Press, New York).

[18] Nobeoka, K. and Cusumano, M., 1995, Multiproject strategy, design transfer, and project performance: a survey of automobile development projects in the US and Japan, IEEE Transactions on Engineering Management, vol. 42 no. 4.

[19] Petroski, H., 1992, To Engineer Is Human (Vintage Books, New York).

[20] Pisano, G., 1996, The Development Factory (Harvard Business School Press, Boston).

[21] Reinertsen, D., 1983, Whodunit? The search for new-product killers, Electronic Business, July, 62-66.

[22] Sabbagh, K., 1996, Twenty-First Century Jet: The Making and Marketing of the Boeing 777 (Scribner, New York).

[23] Scott-Morton, M., 1967, Computer-Driven Visual Display Devices (Unpublished Doctoral Dissertation, Harvard Business School.)

[24] Senge, P., 1990, The Fifth Discipline: The Art and Practice of The Learning Organization (Doubleday, New York).

[25] Simon, H.A., 1969, The Sciences of the Artificial (MIT Press, Cambridge).

[26] Smith, P. and D. Reinertsen, 1998, Developing Products in Half the Time: New Rules, New Tools (Van Nostrand Reinhold, New York).

[27] Stalk, George Jr. and Thomas Hout, 1990, Competing Against Time (The Free Press, New York).

[28] Terwiesch, C. and C. Loch, 1996, The role of uncertainty reduction in concurrent engineering: an analytical model and an empirical test, Working Paper No. 96/17/TM, INSEAD.

[29] Thomke, S., 1997a, Managing experimentation in the design of new products, Working Paper No. 96-037, Harvard Business School (in press Management Science).

[30] Thomke, S., 1997b, The role of flexibility in the development of new products: an empirical study, Research Policy vol. 26 no.1.

[31] Thomke, S., 1998, Simulation, learning and R&D performance: evidence from automotive development, Research Policy vol.27 no.1.

[32] Verganti, Robert, 1997, Leveraging on systemic learning to manage the early phase of product innovation projects, R&D Management.

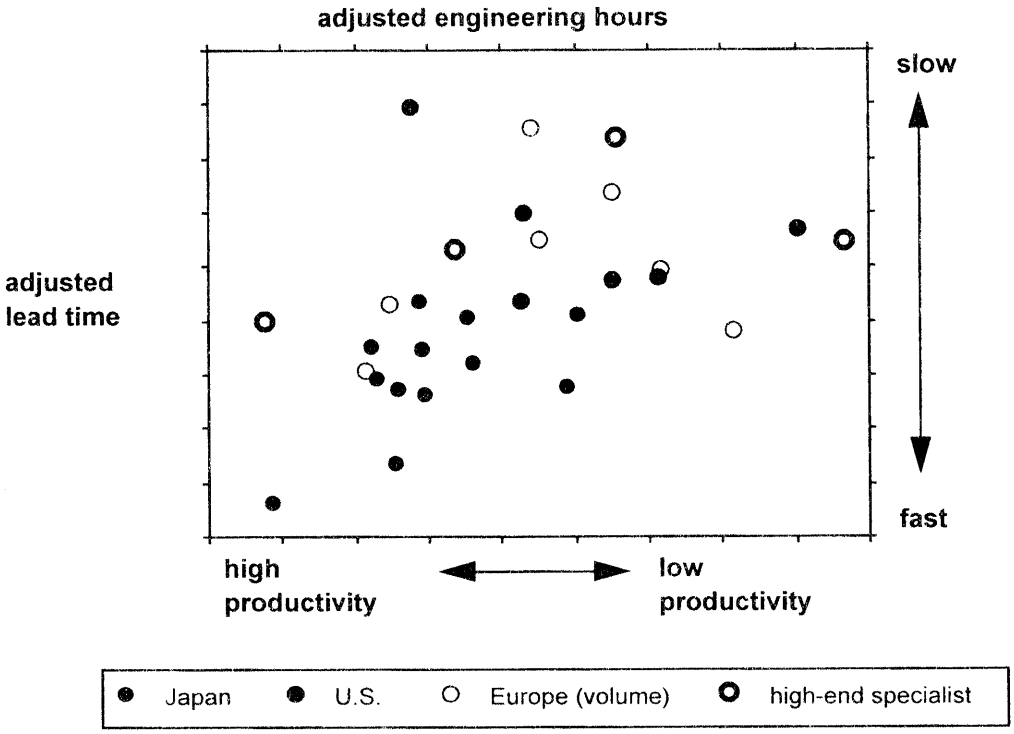
[33] von Hippel, E., 1994, "Sticky" information and the locus of problem-solving: implications for innovation, Management Science vol. 40 no. 4, April.

[34] von Hippel, E. and M. Tyre, 1994, How "learning by doing" is done: problem identification in novel process equipment, Research Policy 19, 1-12.

[35] von Braun, Christoph, 1990, The acceleration trap, Sloan Management Review, vol. 32 no. 1, 49-58.

[36] Watkins, M. and Clark, K., 1994, Strategies for managing a project portfolio, Working Paper, Harvard Business School.

[37] Wheelwright, S. and K. Clark, 1992, Revolutionizing Product Development (The Free Press, New York).



Note: To adjust for product complexity, projects are plotted as residuals with 0.5 million hours/segment (horizontal) and 5 months/segment (vertical).

Figure 1. Adjusted development performance (lead time and engineering hours) by regional groups (1980s) (from Clark and Fujimoto [6]).

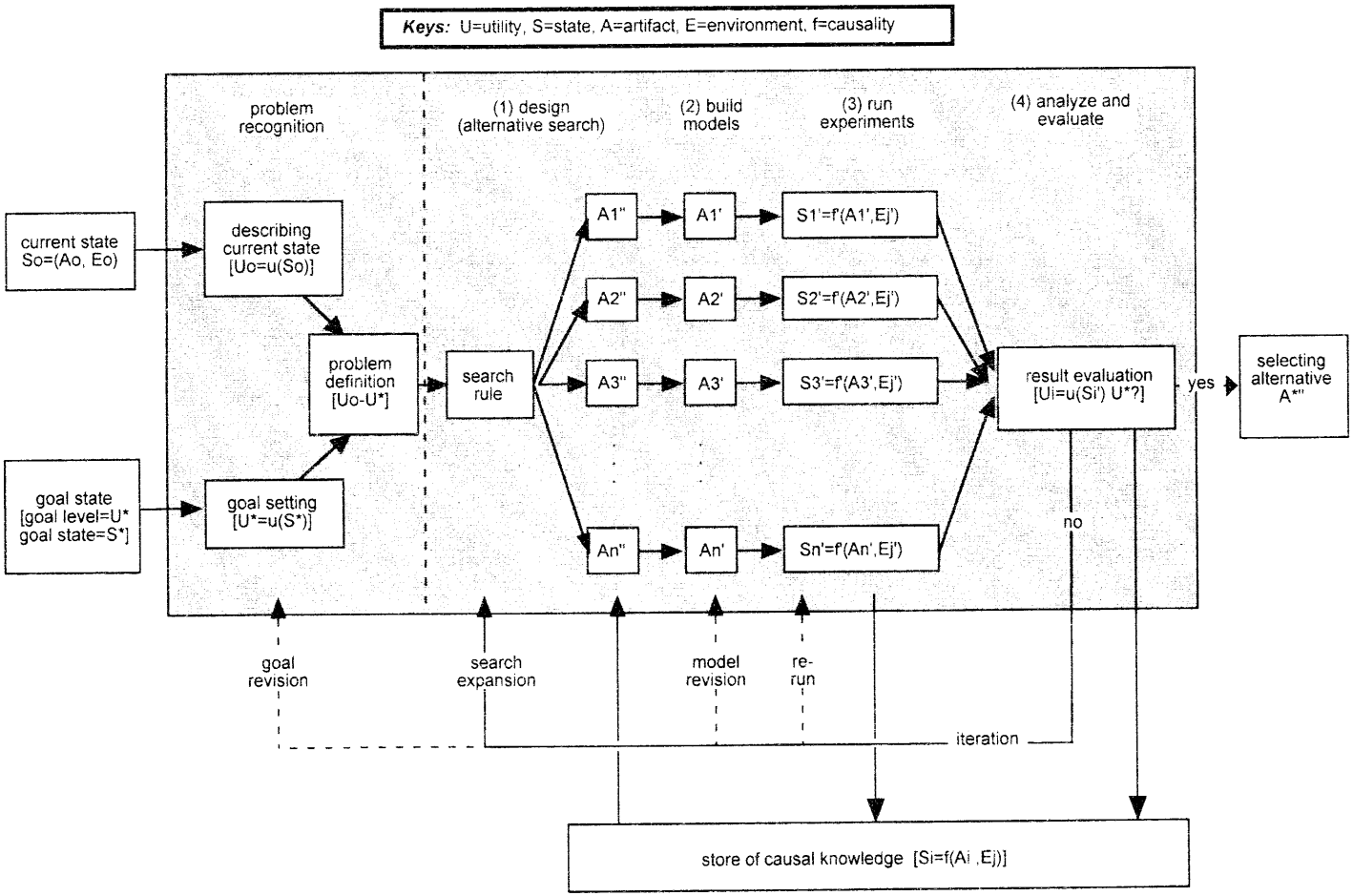


Figure 2. Product development as a process of repeated problem-solving [10].

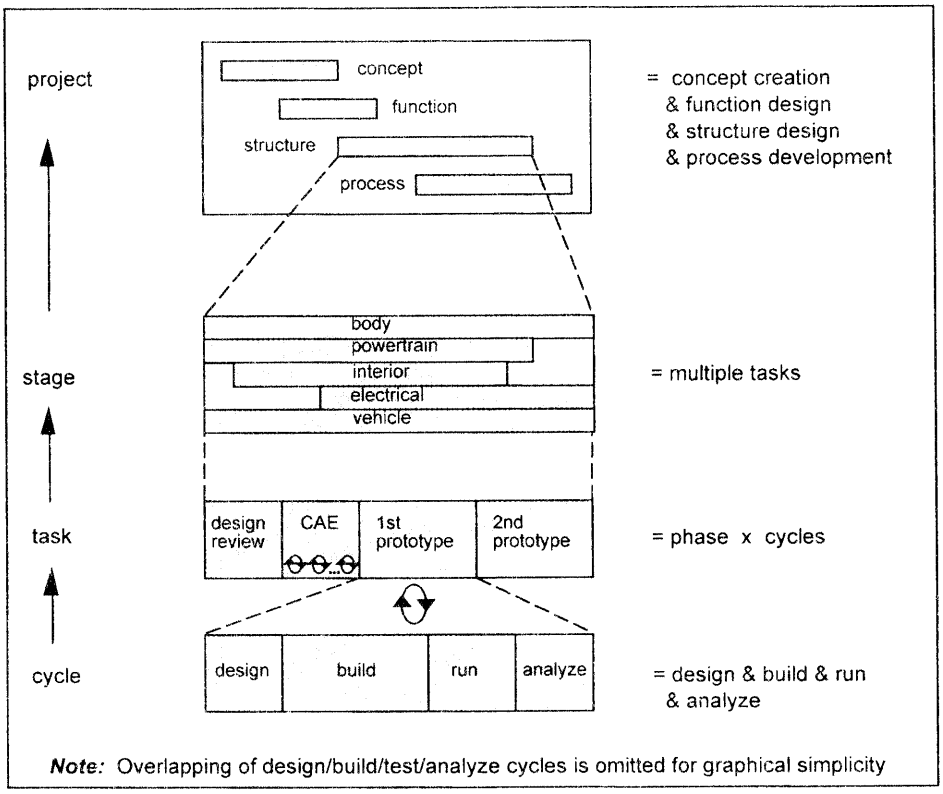


Figure 3. Automotive development as problem-solving cycles and stages.

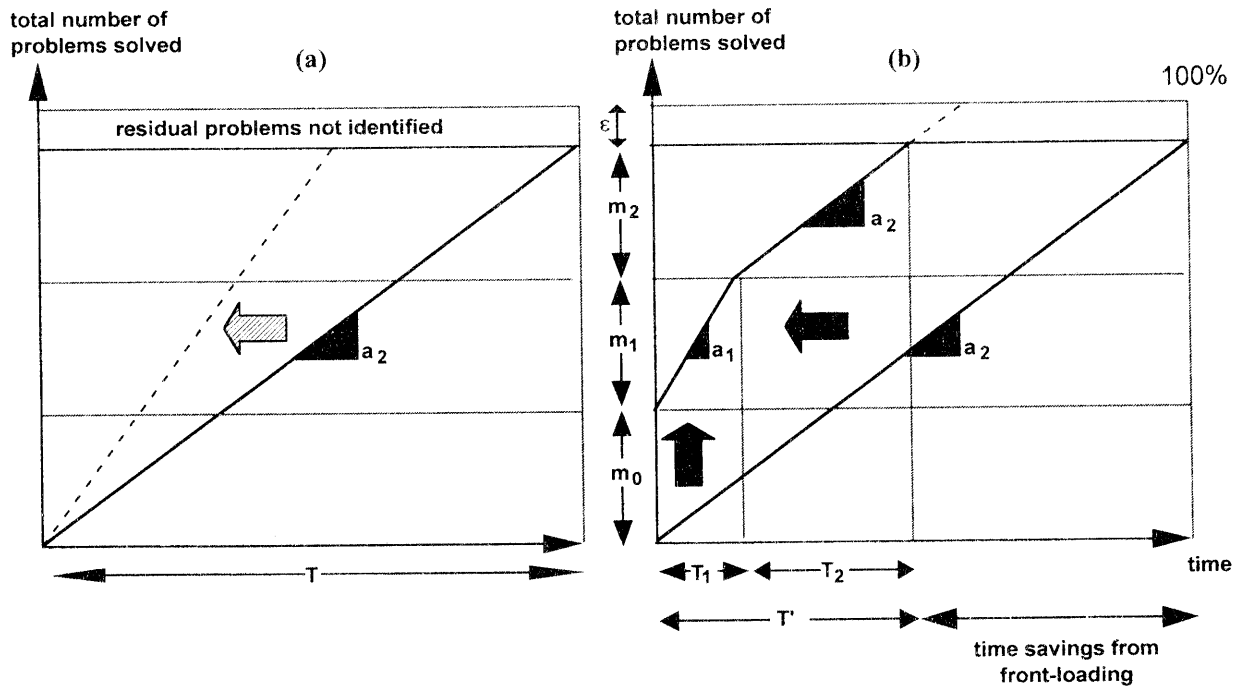


Figure 4. Problem-solving with a single prototyping mode and no project-to-project knowledge transfer (a) and two prototyping modes with knowledge transfer (b). In case (a), front-loading can only occur if the rate of problems-solved (a_2) is increased without sacrifices in fidelity. In contrast, case (b) has a second and faster mode with lower fidelity available (a_1) which can be used to shorten time. Problems where information already exists prior to the start of the project can be identified and solved through more effective project-to-project knowledge transfer (m_0).

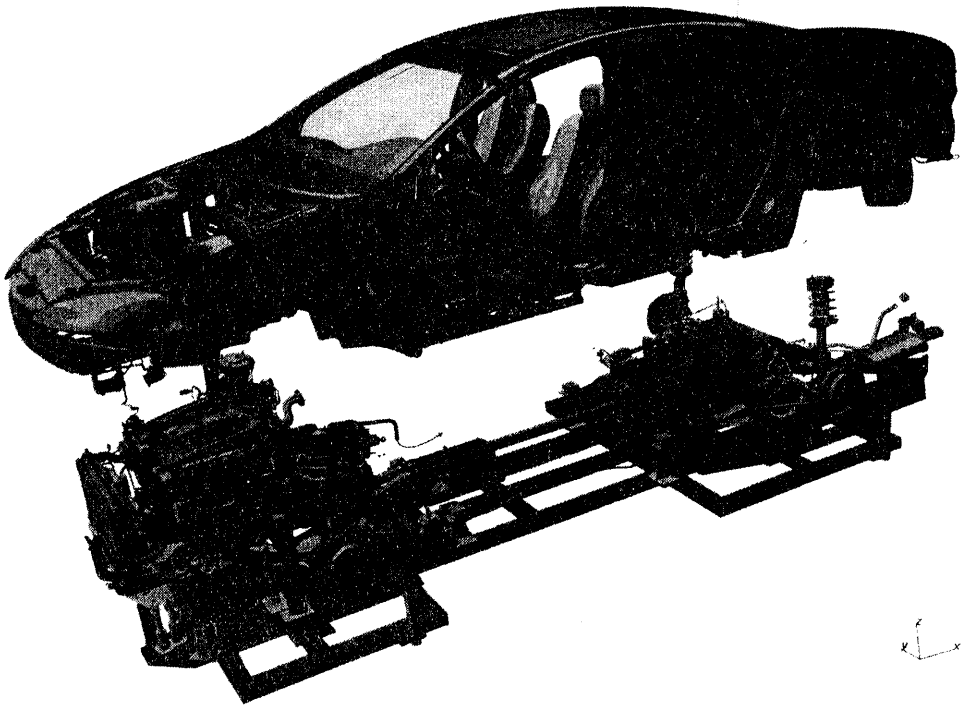


Figure 5. Illustration of decking. Interference problems are identified when the powertrain and its related components are inserted into the upper vehicle body during assembly. Virtual decking with the aid of "digital mock-ups" has helped Chrysler to identify and solve these interference problems before physical assembly takes place.

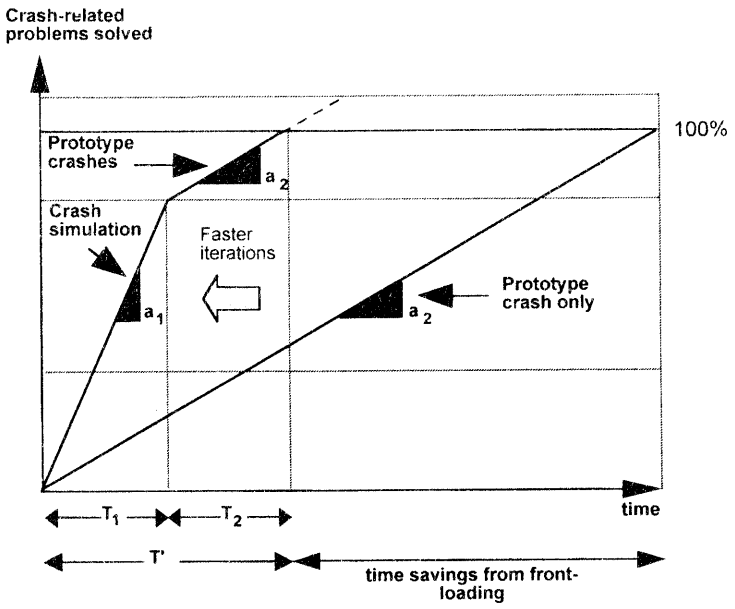


Figure 6. Crash simulation can iterate more quickly and thus increase the rate of problem-solving. Time can be saved by combining simulation and the crash testing of physical prototypes. Note that the dotted line indicates that simulation can sometimes go beyond the identification of problems which could be identified using prototypes only [30].

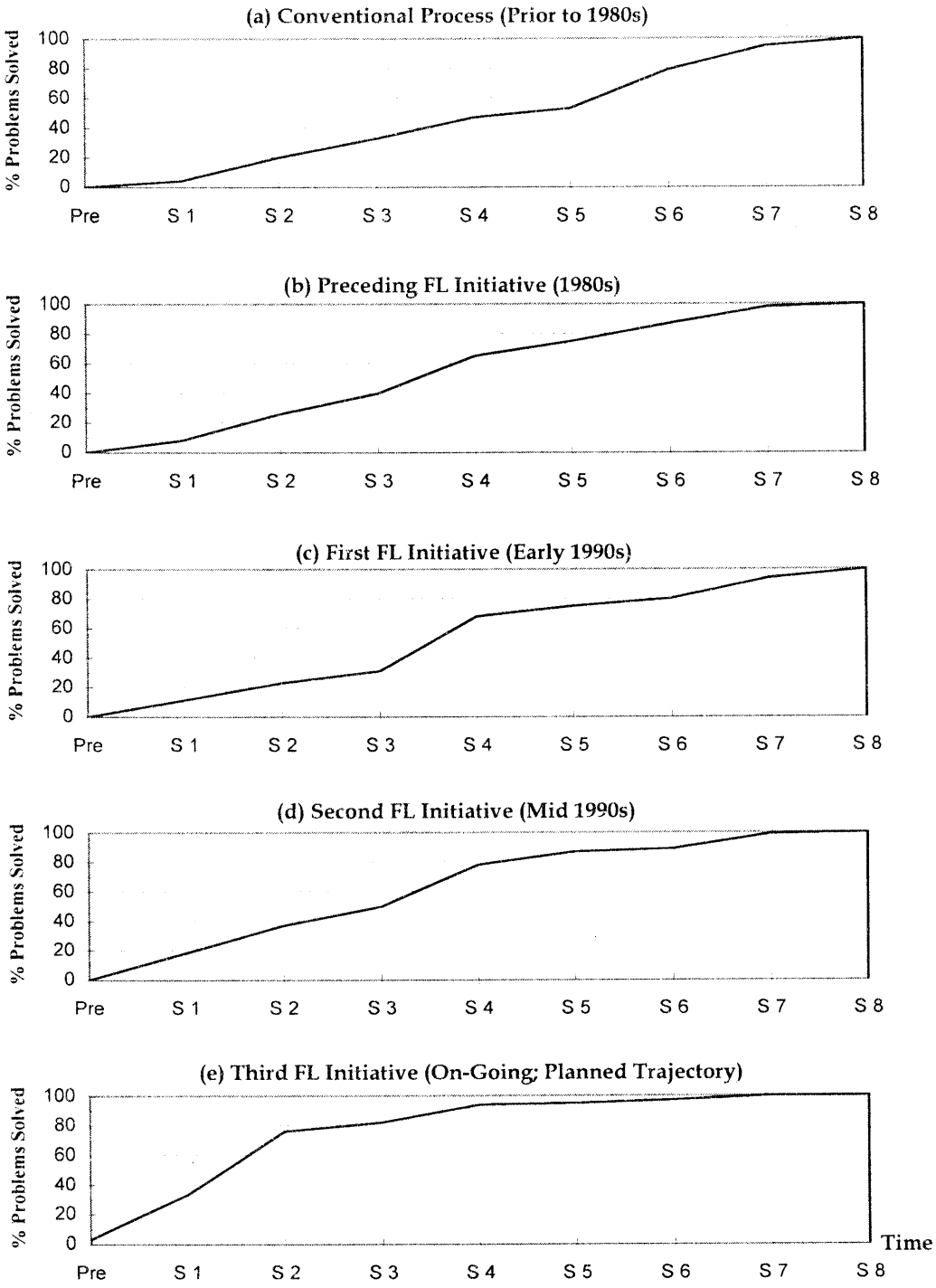


Figure 7.(a)-(e). Problem-solving trajectories as a function of development stages and front-loading (FL) intensity (increasing from "conventional" to "third FL initiative"). Development stages S1 through S8 are defined as follows: (**S1**) structural drawings (detailed layout drawings); (**S2**) prototype drawings (detailed engineering drawings for first prototypes); (**S3**) first engineering prototypes; (**S4**) revised engineering drawings (tool drawings); (**S5**) second engineering prototypes; (**S6**) first pilot vehicles; (**S7**) second pilot vehicles; and (**S8**) job 1 (start of production).

Problem-solving step	Simulation[1] (per iteration)	Physical Prototype (per iteration)
(1) Design	*Technical Meeting* - < 0.5 days	*Planning and Piece Part Design* - >2 weeks (involves many meetings)
(2) Build	*Data Preparation and Meshing* - Small change: <0.5 days - Significant change: 1 week - Entire automobile: 6 weeks	*Design and Construction* - Using existing model: 3 months (at $150,000 per prototype)[2] - New model: >6 months (at $600,000 per prototype)
(3) Run	*Crash Simulation* - 1 day (varies with computer hardware) @ $250/day	*Crash Physical Prototype* - 1 week (includes preparation of test area)
(4) Analyze	*Post-Processing and Analysis* - < 0.5 days	*Data Preparation and Analysis* - 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 (incl. effort):	**about < $5,000**	**about > $300,000**

[1] 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 two months but is not reincurred during design iterations.

[2] 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.

Table 1. Approximate lead times and cost for design iterations using computer simulation and/or physical prototypes (from [26]). Crash simulation allows for a much higher rate, and at a lower cost, of problem-identification and solving than conventional prototype testing.

	Development Cost		Number of Prototype Vehicles	
	Before	*After*	*Before*	*After*
First front-loading initiative	100%	70%	100%	40%
Second front-loading initiative	100%	65%	100%	55%
Third front-loading initiative (expected)	100%	50%	100%	35%

Table 2. Reported improvement of selected product development phases as a function of front-loading activity. (Lead time reductions: after the third initiative, lead times were between 30-40 % shorter than the 1980s schedule.)