Lecture Text

Professor Stefan Thomke

Experimentation Matters:

New Opportunities for Innovation

(*edited for clarity*)

Introduction

What I wanted to do before I get started with the talk is to give you some quick background on myself. Again, my name is Stefan Thomke. I'm originally from Germany, which is where I grew up; graduated there from high school. I did something really unusual for a German. Challenging my parents, I wanted to do something radically different: I studied engineering. And I was trained originally as an electrical engineer. Then later, I ended up getting my Ph.D. in a joint program between electrical engineering and management, over at MIT. Spent some time in consulting at McKinsey, and I joined the faculty in 1995. So I've done lots of work, but primarily in the area of product development innovation management, which is what I'm going to be focusing on today.

But I want to show you something that I think is quite novel. It's been getting quite a bit of attention right now in the field of innovation management. And I'll show you some of the things that we have discovered, learned through our research, and give you a feeling also of how the field is changing as well, or the practice of innovation management is changing.

Growing Through Innovation

So let me quickly get started here. If I woke you up in the middle of the night and asked you what would be your dream, assuming you're dreaming about business—that's a requirement—what would you dream? And probably these kinds of things would be on the list: a steady, predictable stream of breakthrough products, higher margins, rapid development.

And if we spend ten, fifteen minutes, we could probably make a relatively long list of things that would be on there. The problem is everyone knows this. They've heard it many times before. And the competitive advantage doesn't come from knowing these things. It actually comes from being able to translate them into action, being able to execute on them. That's where some of the challenges are in terms of trying to see these through.

Now, what do you think in your own experience—many of you probably deal with innovation, in general; maybe not in manufacturing industries, but service, other industries—what do you think are the challenges in innovation management? Why is this so difficult? Why is every company always concerned about this? Why can't we just say, this is how you do it; let's just deal with it, and forget about it? If you list priorities, it's always among the top two or three things for every CEO.

And here is a list of the kinds of things that we typically see when we deal with this topic of innovation, or product development, or service development. As we said, short-term focus. This is an interesting challenge. Often development managers or project leaders do not know how to do it. That is, if I made a deal with them and said, "You can have all the resources that you want. Just ask. Any amount of money that you want. You just have to

promise me one thing: that out of this process comes a breakthrough product. It's the only thing you have to promise." I would bet that a lot of people don't know how to get started. They could take the money, but they don't know how to get started in terms of process: What kinds of questions do we ask? Who do we talk to? Where do the ideas come from? How do we test the ideas? And so forth. Huge challenge. And then, of course, these things around risk taking, and so forth.

Why Experimentation Matters

Now, the fundamental problem in managing innovation, why this is so tough, is because it involves uncertainty. This is what innovation is really all about. It's an uncertain process with an uncertain outcome, and it's novel. That is, the more novelty that you have, the more uncertain it is because, if there was no uncertainty, we could just go ahead and make it. There would be no point in developing it. There'd be no point in investing in it.

So the problem is, it's the management of uncertainty that is so difficult, dealing with uncertainty, dealing with things that are unforeseen, and that always happen in development projects. I've never seen a development project—and I've seen many in my career—where things go as expected. There are always unforeseen things that happen. It's just a fact of life. There are always problems going on.

In fact, if I go into a development team—small or large—if I ask them, "How are things going?" they would tell me, "Things are going great. We're right on track. No problems. Everything works just as expected," I know they're lying through their teeth. I would get extremely worried because they're in a state of mind where they're probably going to have a very, very big and nasty surprise at some point. That's not what it's about. And so this is really fundamental.

Sources of uncertainty

But let me just quickly tell you about the different kinds of uncertainties that we deal with all the time. Even though I will be speaking a lot about products, this notion of experimentation and uncertainty really applies to a lot of different settings. I've been, for example, more lately seeing a lot of interest in the area of business development and how the notion of experimentation can be applied to the development of new business ventures.

Here are the different sources of uncertainty that we typically deal with. Four categories: Technical uncertainty. That is, does it work as intended? Is it fast enough? Is the fuel consumption right from a functional perspective? Production uncertainty. That is, just because something works doesn't mean that we can produce it effectively and at high quality. So there's a question about whether whatever these people create can be effectively produced at high quality.

Need uncertainty. This is always a big one, right? That is, just because we can create it and produce it, does it actually meet a need out there? And then, of course, the fourth one, the big one is the market uncertainty. Is there a market for it? Does the market size justify the resource investment?

Now, the dilemma here is that when you're developing really novel products markets may not exist yet. So how do you go out and determine whether the market size actually justifies the investment if the market doesn't exist yet, or if the product in a sense creates the market? Very, very difficult.

I mean, there are lots of examples that we've seen recently. We've seen them in cars. Toyota Prius essentially went into the market that didn't really exist yet, because there was no experience. And to some extent, really novel products such as the iPod initially came in and really created a new kind of market. So how do you evaluate something like this at the onset of a project, when you're trying to create a new product, a new product category that creates the market along with it?

Resolving uncertainty
Now, there are fundamentally three ways of dealing with this uncertainty on a daily basis because, no matter what you do, you'll be facing or you face one of these uncertainties almost on a daily basis. There are three ways to deal with it. Of course, experience, ideally, we've done something like this before. We applied our lessons learned to this. But the problem is, if it's something very novel, chances are our experience is actually rather limited here. We may not know what to do.

First principles: This is sort of the dreamer in me. Wouldn't it be nice to have the equivalent of Newton's laws of management? We have a few fundamental laws. We plug in a few equations. The answer pops out. And we don't have to worry about all these messy things, right? We know exactly what to do in a very fundamental way. Unfortunately, it doesn't work this way. We're dealing with very complex systems, social systems, and so forth. Very, very tough to do.

So that, in a sense, leaves the last one, which is experimentation. Experimentation, to try things out, is really the fundamental way. It's sort of the most effective way of dealing with these different uncertainties. Of course, you're thinking now, "Yeah, we're probably experimenting to some extent." Now quite often, when we sit down and write down some of these experiments, it turns out that experiments become experiments after the fact. That is, we've tried something and it didn't work, so it must have been an experiment. It's essentially an excuse for failing at doing something.

But it's not really an experiment at the outset. It's really not thought out as an experiment, funded as an experiment, and so forth. And as a result, we see things like this. Probably many of you may remember *In Search of Excellence*—a big business book; very much in the early 1980s; Peters and Waterman; a great book—and so what they did is they looked at excellent companies, what they have in common.

And one of the things they found first was that excellent companies around the world, all these excellent companies they looked at, they had a bias for action. That means people don't sit around and talk and talk and talk, and discuss and argue. They pretty much just go ahead and try something and then learn from it. Now, what they found was that the companies that didn't perform so well—and this is what's in blue here—was that most big institutions have forgotten how to test and learn. They seem to prefer analyzing the data to trying something out, and they're paralyzed by fear of failure, however small. And that in some sense is true.

I was at a conference last week. They had a very large program at a company where they rethought their way of doing business development. And what they did is they said they ended up spending probably $5 to $10 million dollars on all kinds of reports about which businesses they should be getting into. But they didn't spend one single cent on trying some of these things out.

So lots of studies, lots of reports, lots of market studies, forecasts and all kinds of things about what may happen if they tried something. But there was no money set aside—no budget, however small, with a small experiment: going in, trying things out, learning from it, getting feedback, and then getting the uncertainty resolved very rapidly. And so, again, it comes down to a lot of things: funding, how to manage it, and so forth.

Edison's innovation factory
Now, one of the nice things about being an academic is that we could go back and study things that to most people appear pretty useless. So I said, "Let's go back and study a really great innovator. Let's go back and study how Edison did this." So what we did was we went and looked at how Edison organized his labs, his processes, and so forth, such that it made his people more innovative. And we found something very interesting. And that is, what Edison did is, when he thought about how to resource it, and how to put people together, he really optimized it around speed of iterations; that is, getting rapid feedback. So if Dougal and I were basically working together in the lab, I'd be the experimenter and I'd be coming up with all these great ideas. And he's the machinist who would have to make the prototypes. He'd put us together. We'd be working across from each other; a table in between. And I would say, "Just make one like yesterday but change that one variable." And he would quickly run away, come back an hour later and say, "It's done." We'd test it and then we'd make another one. And we basically had a really fast way of iterating.

Most places today, when you want to do that, first you fill out a requisition. You get five signatures. Then you get a budget for it. Then you send it to another part of the organization. And then you basically get some feedback saying, "Yeah, we'll have it about a month from now," and so forth. And it really slows you down.

Now, Edison's biggest fear, the very biggest fear that he had was that an idea would grow cold. That is, someone had a great idea for a new experiment, a new way of trying something out—and you had mentioned new material—and he or she would go ahead, wanting to try that out, and then no feedback; no way of trying it out. And basically, he believed that if you wait for one or two days or so, people would lose interest in it. They'd go on about their normal lives. They have other work to do and they'd just give up.

So he said, "We've got to do something that allows us to be really, really fast because we never know what we lose along the way if people just don't want to try things out." So I think this captures it very nicely: "The real measure of success is the number of experiments that can be crowded into twenty-four hours." And he really believed it. And he ran his place like this—fast turnarounds, fast feedback on anything—which allowed him to try out many different things.

Technology potential
Now, the reason why it's so much more fun to be alive today than during Edison's time, among many, is that new technologies such as—and some of you may have used them before—computer simulation, modeling, rapid prototyping, lots of new technologies in pharmaceuticals, high-throughput screening, they're all sorts of new technologies that have made it a lot easier. And when I mean a lot, I mean a lot; that is, by orders of magnitude: easier to run experiments, to try things out.

And as a result, two things have happened. The first thing, of course: the productivity of experimentation, in a sense, or productivity of R&D, is changing, allowing us to actually aim at much more difficult problems, much more challenging products.

But the second thing, which I think is even more important, is what I call the "what-if" experiments. We can suddenly ask questions that we couldn't ask before.

What if the following happened? I'm designing a car. If the driver is intoxicated, how would my safety system, or the computer in the car, respond to that? Very difficult to try out in real life for all sorts of reasons.

But now I can do it in a simulator. I can try these things out. I can ask fundamental questions that I just couldn't test out before. And by making it relatively inexpensive, I'm also making it economically feasible to answer these kinds of questions.

That is, before that—and I'm going to give you the example of autos again later, where crashing a car is about a quarter million dollars or so to destroy. Now, when things get so expensive, you tend to amortize lots of things. You throw together a lot of experiments into a single trial just to make sure it doesn't get too expensive. And then, at the end, you don't really know what you learned from because you have too many variables that you're changing at the same time. So as a result, you don't really learn that much from these things.

Revolutionizing experimentation: The electronic spreadsheet
So these things are being adapted all over the place in lots of different industries. And many of you have probably seen this before. Here's one that everyone has used before. And I just wanted to raise this because this actually ties it back to HBS: electronic spreadsheet. Can you imagine there was a time when these things were not available, when you had to do this by hand? Oh, you remember. You did it by hand? It must have been an awful time. That was very, very difficult. Now, Dan Bricklin, who graduated from here in the MBA program in 1978, told me this story of what it was like back then in the classroom here. And some of you may know yourself. So Dan told me the story.

He did this in the first year. He got started on it in the MBA program. He worked on it with his friend Bob Frankston. They were both undergraduates at MIT. They remained friends. Dan was here in the MBA program and I think Bob had a job, and they worked on it mostly at night.

And it was also relevant. Dan's background was word processing, so he had some experience with simple word processors. So he said, "You would walk into the classroom and you would, for example, do a finance case. The professor would walk in and fill up all the boards all the way around. After eighty minutes, you're done with all the calculations." He said, "The answer was always about 12 percent." And he'd be sitting there, daydreaming, and saying, "What if I had a helmet like a flight simulator, and I could basically change one of the assumptions, and quickly the new answer would pop out." And he started to dream along in class of what could be possible, sort of a word processor for numbers. That was the vision.

And then he basically went ahead and changed it, and they developed it. Unfortunately, software patents didn't exist back then, so he was about two or three years off. He said, "If it had been two or three years later, his name would be on a building here." Now, we just ask him to come to class.

In any case, this is a great example of essentially a simulator for numbers because you can do very high-speed financial simulations. You change an assumption, ask a question. Quickly, if you've got the model in the system, the answer pops out. Now imagine having

something like this for all kinds of different problems, very difficult development problems, having essentially sort of a simulator like that available. And that really is what it's all about.

Crash simulation and modeling at BMW
Now, here's another example about cars. I just wanted to quickly show you because today I will be using some examples from the auto industry. This is a research project we did on BMW—very fascinating. Crash safety, by the way, is the most challenging problem in automotive development. Very expensive. "Finite elements" is the level of granularity of the model. So you want to have more finite elements because the model will be more representative of what's really going on in the world.

And so this is just a difference. Three thousand—this was in 1982. Now they're going up to probably about one million elements, doing it in a small fraction of the time that it was done back then, where it took them almost three months to do something very simple. It's quite fascinating. This is in part the reason why we have much safer cars than we had, say, ten, fifteen years ago, because it is really fundamentally changing the way we think about automotive design.

Technology-performance paradox
I could stop here and say these are really exciting times—lots of great things going on, and so forth. But there's another side to this story as well. And we discovered this as part of a very large research study that we've done here. We have an automotive project, here at the Harvard Business School, that's been running for nearly twenty years.

It was actually started by our [former HBS] dean, Kim Clark, with his graduate school student, Takahiro Fujimoto, who is a professor at the University of Tokyo. And they started a very, very large study here called the Automotive Development Study. And a lot of their practices, best practices in product development, came directly out of that study.

So we've been continuing with this study, going back to companies, collecting data. And we collect a lot of data. We typically go in and collect data at the project level. So we look at automotive projects, lots of platform projects. Just to give you a sense for it, a typical automotive platform project is about a billion dollars, give or take a few, in terms of expense. So there's a lot of money at stake. So we collect a lot of data at the project level about how the project's being managed, about process, about performance. And then we want to link some of these best practices to performance variables.

So we've been tracking this over time now. And we have more than seventy projects in our database that we've collected over time from most car companies around the world. So this is quite rich because it allows us to look at these drivers of performance by region, by company, by project, and allows us to do wonderful things in terms of explaining why certain projects are more productive than others, or why certain projects are a lot faster in terms of time to market than others. Quite fascinating.

In the first study, there was a big gap that was really fundamental here between mostly Japan and the United States and Europe, where the Japanese projects were probably about twice as productive. So we studied that and we found that in the early '90s, the performance converged, became much closer to each other.

And then when we studied in the late '90s, we saw divergence again: The Japanese projects that we studied were much more productive in terms of performance than some of the U.S.

companies that we saw. And usually product development is an early indicator about what's to come because these are projects or cars that are in the pipeline. So usually you see the fallout years later.

Then, as we said, we looked at these practices that accounted for these performance differences. And when we looked at the practices, we saw that a lot of the practices in fact have converged. So a lot of the companies have in fact learned. So I said, "I know the answer. I know why these companies are more productive than the others. It's new technology. It's computer simulation, three-dimensional modeling, and all these things. So the companies that are more productive must be using a lot of that sexy new technology." That's at least what I *wished* I would see.

So we went and we looked at the projects and it wasn't true. So we looked at the projects and we found that especially some of the U.S. companies that were not doing so well in terms of overall productivity and time to market were actually using the leading technology that was in the field. They were writing the biggest checks. Senior management was writing checks for these technologies. And this is expensive stuff. And then, lo and behold, when we looked at the payoffs, we didn't see the payoffs in terms of performance.

On the flip side, we saw other companies, especially Japanese companies that were much more careful about the spending but paid a lot more attention to integration of the technologies, getting a lot of performance out of it. So that created what we called a technology performance paradox that's inside; that it wasn't really the technology investment, even though it was an enabler driving this, but it was the degree to which the technology was integrated into the processes, which meant in fact that processes had to be changed. Organizations had to be changed. And the whole approach to product development had to be changed as a result of that, which was a very different philosophy than basically writing a check and saying, "Here's a new technology. Go in and save some money."

We had one company in our sample, which I don't want to name, that was really, really wonderful in a lot of different dimensions and wonderful in a very questionable sense. These prototypes, I told you, are really, really expensive. So one of the underlying reasons for spending all this money is that "we could save a lot of prototypes."

Typically in these projects, a company may build 100 to 200 full-scale functional prototypes. Now you can add the numbers. If it's a half a million dollars or so, this adds up pretty quickly. So the idea was we use these technologies and, as a result, we can reduce the number of prototypes significantly. And that's how we get the return on the investment.

So we have one company that was by far the biggest spender on these technologies. They also at the same time had the most prototypes in the whole sample. So something didn't quite work. There was supposed to be this substitution effect. They spent the most money but they also had the largest number of prototypes in the sample.

So we went in and we presented the data to them, and then said, "Congratulations. You win in both dimensions. You've got the best technology and, at the same time, you also have the largest number of prototypes. Please explain." We were curious.

And so the chief engineer, I remember at the time, she sat there and she listened to this. And she said, "Oh, I know what happened here. It's totally clear. I know exactly what happened. We basically didn't want some of these technologies. But management thought

that this would be a great way to save money, and so they asked us basically to introduce some of these technologies. So we introduced the technologies but we didn't really trust these kinds of things. We didn't trust simulation models because it wasn't real."

"And as a result, each time we did a simulation, we built a prototype to check to make sure that it was actually accurate. And so the more we simulated, the more we modeled, the more prototypes we built just to always verify that the computers were accurate. As a result, we ended up building a lot more prototypes than we used to." And so there was no cultural acceptance, in a sense, of the technology. No one has thought about it. They thought we'd just drop it in. And all the engineers love technology. They would just suck it up like nothing, okay? They'd just be happy to have all these tools at their disposal." That's actually not what happened. And so we had some practices in terms of how they were integrating it. It looked very different than what we saw in some of these companies.

Organizing for Rapid Iteration

What I want to do now is share with you some of the other things that we've observed, some of the practices that we observed that allow you to use some of these technologies a little more effectively, and to be a more effective experimenter; to use the power of experimentation and innovations more effectively.

Learning by experimentation

So let me just quickly throw out a model to you. This is probably a useful way of thinking about experimentation cycles and the role of experimentation in large developments projects—to think of a four-step process. And this is, by the way, not just true for product development. You could think about it this way also for a business development perspective or service development.

So we have a first phase which I call here very liberally "design," but this is the phase in which new concepts are being created, new ideas are being created. Brainstorming happens. New experiments are defined. New trials are defined. We've got the second phase where we've built some kind of a model because we have to try it somehow at some point.

So we built some kind of model. It could be a prototype. It could be a computer model that we saw earlier. It could be something as a physical prototype. It could be a mental model. It could be a business model. It could be some kind of a model that's supposed to mimic the idea that you've come up with. Then we run it. We collect data. We analyze in terms of what happened and what we learned, and the cycle continues. In a large development project, these cycles can happen thousands or tens of thousands of times because we go through many iterations to try things out.

Now, what's important here, and that's why I showed it as Department *A* and Department *B*, where the challenge comes in is that sometimes these steps are done across organizational boundaries. So what we may find is that, if you are in an engineering organization, the design engineers are the left-hand side. They're in Department *A* and they're basically designing stuff. And the people who are running the tests—the test engineers, the test department—are on the right-hand side. And they're sort of divided up. They're in different organizational units. They have different managers, and different processes, different objectives, and so forth.

And then the management across these boundaries can be quite challenging, extremely challenging, especially when you're trying to do what Edison did, and that is to accelerate

the speed of iteration, the speed of trying things out, and then to try to bridge these different boundaries. Very, very tough.

So to give you an example, a few years ago, a manager from a European pharmaceutical company—he's the head of Discovery—came to see me. I've known him from some earlier times, some other things that we've done together. Anyone who's here from the pharmaceutical industry, I apologize profusely. I'm going to simplify a great deal now because I'm trying to describe something that is actually very complex.

And he said, "I've got a lot of scientists working for me. And their job in a sense is to come up with new concepts for chemical drugs, for new compounds. So we want them to be creative. We want them to think outside the box, think about new disease mechanisms and possible compounds that could be potentially used for drugs. So we've done a lot of things. We've done all the brainstorming. We've done all these things. And my people have come up with wonderful things that we can test out."

So we take all these wonderful things, and we move over, and there's a different part of the organization. These, by the way, can be fairly large. In autos, for example, the testing area is about 20 percent of an organization. It can be very, very large.

So we come over to a place called "animal testing." And they have small and bigger animals, very small animals and very big animals. And these animals are essentially models of human organisms and, in a sense, prototypes. And they want to test out some of these molecules on these animals and then get some feedback about the effect, the potential effect these new compounds have on these animals.

So he went over and told his counterpart, "I've got all these ideas. And here they are. We'd like to run some tests." They said, "We'd be happy to do this for you. We're a service organization. We're here to do these tests and you can expect to get some feedback in probably around three to four months. We've slotted you in. You'll get a nice 150-page report."

And he said, "All I care about is a small test that takes around four to five days to run. I just want to run it on some rats or some small mice. And I want to know at the end of the week whether these things are alive or not. And I want to get one page back, a graph that just shows me what happened to them. And that's a really simple test. So these people want me to wait for three and a half months, and then they'll write a 150-page report. It's irrelevant to me. By the time I get their feedback, I'm not interested anymore."

So one basic problem we have is that these interfaces get in the way.

The impact of interfaces

This is, I promise you, the only academic slide that I will show today. So this is a result that comes from a field called "queuing theory." For those of you who are in telecommunications networks, you've seen this slide many times probably. It says something very simple. It says when you take a resource, any kind of resource, and you start utilizing it, in a sense loading it up, and you have different jobs or tasks that arrive in a random order, the closer that you get to 100 percent in utilization, then the waiting time will basically go to infinity. Now that doesn't actually happen so late. What you see here is that starts picking up when you hit around 70 percent. So, once you get over 70 percent utilization, things tend to explode very quickly.

So here is the problem in my little example here that, on this hand, percent of resource utilization, I've got my testing center, animal testing. So they're a service area. They're run as a cost center. What do you think the objective of the manager running this cost center is? Utilization, asset utilization, resource utilization. Typically we have this when we have central IT managers who look out for the equipment. When you ask them, how are things going? They'll say, "Well, we've got a great month. We're getting utilization up to 90 percent. I think we're shooting for 92 percent next month." And so their lives revolve around getting this resource utilization up and up and up because that means they can squeeze more performance out of the cost that's invested into the resource because that's how they manage: To what extent can we utilize these cost centers?

Now up here, we've got the Discovery folks who basically care about getting fast feedback. They come up with new ideas and they want to get fast feedback on their ideas. So they care about waiting time. This is the stuff that Edison really cared about. They want this to be close to zero. Because I have an idea, a new concept, I design it. I want to have instant feedback so I can continue. I don't want to sit and wait around for weeks and then see whether this worked or not because I do many of these iterations. And so we basically have already a conflict built in, different incentives across these interfaces. And this is probably true in many places, test centers, all kinds of cost centers. And as a result, this has to be overcome right away.

I suggested three ways to overcome that. One is alignment of incentives. So we need to make sure that we care about this. This is what we care about. We're not in the business of doing resource utilization. We're in the business of developing new things. So to get this down we realign incentives. So we ought to measure something that these people care about, such as response time.

There's a resource solution to this. We can take advantage of the steep slope of this curve here. That is, if we just spend a little bit of money on getting more resource, we can get a fairly big bang in terms of getting the time down. So sometimes this little spending up here results in a big improvement down here.

And the third one, of course—that's what he wanted to hear—is the organizational solution, which is basically take all these and put them under one person. And then basically run and make sure that these incentives are aligned. He cared about getting these drugs through the system as quickly as possible. And he wanted to minimize the interfaces as much as possible.

Rapid iteration and field research
Let me give you another example. Anyone here from the auto industry? Okay, so halfway. One of the big problems right now is that when they design cars, engineers do a lot of the design but they don't actually do the computer drawings. They have people who are called CAD operators. These are operators who do drawings on computer systems. They draw three-dimensional pictures based on the information that the engineers give them.

The problem is that all of these CAD operators, and there are many of them, sit in front of their terminal and they typically sit in different buildings. They don't sit next to an engineer. They sit in a different building. They're highly unionized. And the engineers are not allowed to talk to them. Yeah, it's surprising, but it's true. They're not allowed to talk to them.

So if an engineer wants to make a change to an engineering design, they typically have to go through some central resource that creates a work order. And then the work order gets

sent to the person who is basically operating the CAD—computer-aided design equipment. And then that person, once they get their job order, makes the change and then they send it back to that central person, and then all the way back again. A very, very complicated process with a lot of overhead.

And the CAD operators won't touch it until they have instructions to do this. Now you can imagine the enthusiasm that engineers have when they want to make a small change. They're sitting here: "Do I really want to go through all this crap or do I just sort of leave it?" And also, the speed or the slow speed that's involved when you're trying to make a change like this. So this is really the essence. These are the interfaces that are built in, unfortunately, in the system, that make these cycles really, really slow. And it gets in the way of innovation and it gets in the way of experimentation.

Projects as organizational experiments
Let me just give you a quick example of what I call an organizational experiment that we followed at BMW in Germany. And here on the left-hand side—I told you earlier about safety design—we've got a standard process where we have these cycles. We've got engineering, we've got prototyping here. It's about a quarter million dollars or more per iteration. If you want to build one of these cars and crash them into a wall, typically what you see when you watch these shows—*Nightline*, and so on—they have these tests.

So when they build these functional prototypes early on, it's very, very expensive to make them high fidelity and crash them. And it takes almost half a year to get one of these things built, and then destroy them. So you can imagine there's a great deal of reluctance to try these things out unless you really have to. And in spite of that, they still have to build many of them.

Now what they wanted to do is they said, "Let's try something different. As a small organizational experiment, not in terms of rolling this out to the rest of the organization, but let's try it out to see what happens." So they went in and they said, "Let's put all these people together. Let's put all the decision authority, in terms of how to do this, let's put it together into a small cross-functional team. Get rid of the organizational boundaries." And they wouldn't have to go and ask anyone whether they wanted to make these changes. And we also support them with some of the new equipment, new simulation equipment, where they could do this very quickly. And at the end, we'd give them a budget where they can build physical prototypes to verify the results.

So they did this. This is what happened. They would get together on Monday morning. They had some problems at the beginning, actually, getting everyone to the same table because before that, they were all living in different worlds. So you get them together in the morning, they brainstorm, and then they would go out. They would run some of these tests. And then they'd take the results, go back, review them.

And the next Monday, they would meet again and then decide on what sort of next test to run, and so forth. So that cross-functional team: very mixed. And they ran ninety-one experiments over this time. When they started to do this, some really, really interesting things started to happen. It looks pretty straightforward here. Let me give you two examples.

First example: One of their safety engineers went to a conference. They have these conferences where different safety engineers from different companies get together and they present new concepts, new safety concepts. So he went to one of these conferences,

sat down, and the safety engineer from a Scandinavian car company that's known for safety, but that I can't really name on camera here, presented this to them. And they said, this is very interesting. The person ran back to the team and said, "Hey, I just went to this conference. I saw this really interesting concept, and what do you think?" And then someone said, let's just try it out.

Now, you're saying, sure, why not? But before that, this was a very radical proposition because you would never do this. It's like starting a new project. You'd have to get budgets, you'd have to do all kinds of things, and it would take probably half a year to a year to get any feedback on whether this works or not. But here they've got all the tools, they've got all the decision authority within the teams. They said, "Sure, let's do it."

So they ran it. A week later it got feedback. They looked at it. Not much of an improvement. So they started to discuss it. They started to think about it. They went back to their books. Turned out they had some suggestions on how to make it much better. And within a few weeks, they had actually taken an idea that they picked up somewhere else. They improved on the idea and had something in their hands that was actually superior to what was being presented to them just because they had the tools and the process available to them.

The power of "what if"
The second example is fairly detailed, but I found this to be one of the most interesting ones because it really showed the power of challenging assumptions. They got together again on one of those Monday mornings. And one of the engineers said, "Why don't we just make it on the side here"—that little green area here—"just make the body stronger, just make it stronger?"

And they all sort of looked and said, "Yeah, get out of here. So what's the point? So we're adding weight to it, so fuel consumption will go up? We'll never meet the criteria, design objectives, and so forth. There's nothing novel about it. We know we can make it safer. But then we will basically have to drive a tank. So go away." But he insisted, came back, said, "No, no, guys, we have to try this. Let's just try it out and see what happens."

So they started to argue, until at some point, someone actually noticed that the incremental cost of arguing was higher than the incremental cost of trying it out. And they decided just to say, "Okay, let's do it." So they ran it. So they made it strong on the side, got the results back, and it got worse. It became actually less safe for this passenger here—the dummy sitting here, a virtual dummy—to be in the car.

You have to imagine the shock. So here they are. They've been doing this. Some people have twenty years of experience and this thing comes back. And exactly the opposite happens than was expected. So what do you think they did? "Let's do it again, okay? We don't believe any of this, anyway. So let's do it again." So they ran it again and, unfortunately, no matter how many times they ran it, it didn't change. It just kept on coming back at them.

Then they decided, "Let's go back and start checking our assumptions. Are all the assumptions that we've made as a team, are they really true?" And so they went back and they cracked their physics books and really researched this. And it turned out that for some of these problems, they had overlooked a secondary effect, which happened to dominate in this particular instance.

Just to give you a very, very simple, quick explanation: What they had assumed was that when they make this stronger here, then this vehicle is more likely to collapse further up. And as a result, you have more injuries in the upper region of the body, which is actually worse because you're causing another part of the car to fail. So that the solution to this problem was actually counterintuitive, and that is to weaken the area so it's more likely to collapse down here. So the solution was completely counterintuitive. It was completely the opposite of what they had assumed all the time.

You have to remember that this is a technical thing, but that's one of the powers of going to action, going to experiments very quickly. We live in very complex worlds, and I would bet that when you're running, when you're trying to start new businesses, things are no less complex than in this crash scenario because you're dealing with lots of variables that are unknowns. But we sit around and sit around and discuss and think we can forecast everything. We make assumptions, right? Many of them may not be true but we're convinced that they're true. And the thing that challenged them was to do the experiment because the experiments forced them to face their assumptions. Because the experiment kept on coming back to tell them, it's not what you expect to happen. And they had to go back and they had to revise their assumptions.

So that's part of the power, no matter whether it's in this setting, or whether it's in a business development setting, or a service development setting. Doing the experiment will force you to test your assumptions and will go back and will force you also to change your assumptions. And it can be much more effective than trying to forecast a future that is totally uncertain to begin with in these complex scenarios. So, very, very interesting example, and it completely changed the way they thought about it.

By the way, this relatively cheap side project here ended up improving side-impact safety, which is the most important one. It's also the deadliest one because these accidents tend to be quite fatal, by over 30 percent, which is a huge improvement, given the resources they invested. And most of the value came from the way these teams were organized—decision authority; they faced the assumptions; the computer was an enabler. But they could have probably given that to someone else, assuming it was fast. So big change here.

Now, and this has been pointed out before, when you start doing all this, when you suddenly do faster iterations, getting fast feedback with diverse ideas, and doing it earlier in the development process where things are easier to change, you're going to face a lot more failure. Early experiments are useful because we can eliminate options early before we invest more money. But again, when things are early, when things are less certain, you're more likely to fail. So that's a big problem.

Failures and mistakes

And by the way, I want to make a quick distinction here. It's a bit of a semantic distinction, because the two words are semantically very close—failures and mistakes—but I want to highlight that. And that is, there's a difference to me between mistakes and failures. Mistakes to me are something where no new or useful information is produced. Something that you've already done before, say, in a previous project, you didn't learn anything from it. And you do exactly the same thing again and, not surprisingly, the same bad thing will happen again. Didn't learn anything from it. This is very different from what we call a failure here which is the outcome of an experiment, or can be the outcome of the experiment, because we learn equally or sometimes even more so from failures than we learn from successes. Very different.

So I always use this example. Remember the first time you went to the kitchen when you were small children, and you touched that hot kitchen stove. There was a lot of information coming back at you at this point. Very first time. A lot of learning going on in that one second. The second time you touched it, there was no new or useful information coming back at you. The thing was still very hot. Same outcome. So we need to allow an organization that allows people to fail. At the same time, we want to minimize these mistakes, these repetitions, and that challenge is quite, quite big.

Why? Because people don't like to fail in front of others; as I said, important gaps in knowledge. People may be viewed as incompetent. In projects, we always like to pretend that everything is great, everything is successful because people who are successful get promoted. How many people do you know who have been promoted for killing a project? I've never met someone.

There is this great story about old Watson, T. J. Watson, who built IBM into a sort of global powerhouse. And lore would have it, there's this young manager who just basically blew $10 million for IBM, which is a lot of money today. It was a lot more money back then. Walked into his office, and Watson sat there in a very intimidating manner. Young man walks into the door, says, "Oh, Mr. Watson, before you say anything, I want to apologize. I've just lost $10 million for your company. Here's my letter of resignation. I'm sure you want me to leave now." And Watson looked at him and he says, "Have you lost your mind? I've just spent $10 million educating you. Why would I want you to leave?"

And that's the notion because he probably thought of this as an experiment. It's a new venture. It was an experiment and failure was a possible outcome. So, of course, it would have been nice to be super successful. The objective was not to get rid of people because they failed but to quickly move on to the next iteration, learn from the experiments, go to the next iteration, and then iterate until we make it work, until we get the right formula to work.

And so we've seen this in a number of settings for companies saying, "Let's forget about it. This is a possible outcome. Let's quickly change. Let's learn. Let's go to the next round. Let's go to iteration two, three, four, and then we'll get it to work." And so that's what we need to do to distinguish this, these kinds of things.

Managing uncertainty

Now, we've done a case, IDEO Product Development, which I've taught in my course in the second year—a great company that essentially created a culture around the ability to fail, especially very early in development, where the objective is not to avoid failure but to expose it as early and as quickly as possible. And they do it through prototyping. The way we communicate new ideas is through prototypes, and the way we get failures out of a system is to get the prototypes to fail as early as possible, as quickly as possible.

There are other things as well. 3M also has a culture that's built around failure. They even celebrate failures. And there are some companies who have really done that. They emphasize that. They say, "We don't want to force it out. Now the second time around, of course, we expect you to learn from that. But the first time around, we want you to drive that out as quickly as possible."

Experimenting Early and Often
Now, I want to quickly also spend some time on the second block, and then we'll have about five minutes or ten minutes for questions, if you have any. Experimenting early and often. Here is the idea in product development.

Value of information decreases over time
This is probably one of the most fundamental curves in product development, any type of product development environment. It's probably true for a lot of settings as well, technology development as well.

And it goes as follows. That is, if we break the process up into some kind of a generic process—and I've just put up a very generic process here—different phases where we generate the knowledge and the concepts early on, and then we go all the way down to manufacturing; ramp up to market launch, and we look at the ability to influence the outcome and make the changes, we can make a lot of changes very easily at low cost at the beginning. But as we're going down that funnel, we get increasingly locked in because we make commitments. If we're in a manufacturing environment, we order equipment, we get tools. In other settings, we get marketing all ready. So everyone makes a lot of commitments. The further down we go, the more expensive it becomes to make any changes to what we have done. There's even a word for that in the field of product development.

We call it often the "Rule of Ten," even though it's probably in some environments even much worse. And that is, if we go from one phase to the next, it's at least one order of magnitude more expensive and time-consuming to make changes, even in environments where you think changes are easy. If you look at software environments—and that has been studied over and over again—where you think it's easy to make changes, right? Not true. You go down and you have orders of magnitudes in terms of costs that you incur to make even simple changes in software systems. And you think that's an easier environment; manufacturing: even much worse. Try to change a car in the late stage by widening it by two or four millimeters or so and you're looking at millions and millions of dollars of costs. So this is very difficult.

Now, that of course, creates a premium on early information. That is, we want to generate a lot of information about failure, what works and what doesn't work very early on, because the longer we wait, the less value the information has because we can't do anything about it, other than killing the whole project. So that's a big problem here, because when you start looking at the way resources are being spent, the picture looks exactly the opposite; that, in fact, most organizations spend most of their money down here and actually try to spend very little money up here. And so we have this basic dilemma out here. That is, most of the opportunities are up here but most of the resources are being spent down here.

So that's the idea about what can we do to drive testing, prototyping, experimentation as much as possible to the front end of a process, where the information is most valuable? Because the reality looks like this. And again, very simplified picture of car development, where we have concept definition. Development is when the engineering happens and series development is when we start to do production engineering.

We start here, we have data. We want to build something to get feedback on whether this thing works so we start doing this. But, by the way, when we're shrinking time to market, the freeze decisions—at some point, we have to freeze everything so our people can actually start working on the production process—comes along, but we're too slow, and the

information is not available to us yet. And so a lot of these freeze decisions are made without feedback on some of these things. So we're heading down that road, and unless we somehow restructure this part, it's not going to work. So what happens is, when we get the information, eventually the feedback will come, but we're past these points when freeze decisions are made. These are big decisions for a company. Basically, we don't do anything with them unless there is a disaster. Unless we run a test, unless we take a prototype and we crash it, and it comes back, and the prototype crash tells us, the data tells us, that we're way over any requirements—negatively over—that the government imposes on us because there is a National Highway Transportation Safety Administration that tells us what's permissible or not.

And then panic breaks out. This is just sheer panic. Then everything is unfrozen, and so forth, and the whole process gets thrown aside and it gets really, really expensive. So what happens here is a lot of these things are set up for verification, but not really useful for learning and experimentation in the way we talked about it today. So again, the premium is, or should be, on what happens up here.

Front-loaded product development
This is a concept and it's the last concept that I'll expose you to today. It's what we call "front-loaded development." And the idea of front-loading means front-loading problem-solving, front-loading experimentation and prototyping; moving it to the front of the process, which means sometimes spending more money; having different processes, and so forth, at the early stage of development; more iterative elements than the linear elements that we've seen here before.

And we can do it in three ways: doing lots of fast iterations, à la Edison with new technologies and other means; learn from projects—that is, minimize the mistakes; and then by doing it early, having much more radical things that we can do as a result of that.

Now I want to show you one last example and then we can have some questions as well. Since we talked a lot about cars today, I want to end it with a car example as well. This comes from Toyota, which arguably is probably the benchmark in the industry, both in terms of manufacturing and product development. And what they did is they've lived this concept in a very interesting way.

If you go back ten, fifteen years, they had a whole number of different initiatives aimed at reducing time to market and getting productivity up. This is, by the way, something that every company has. Ever since I've been in this field, every company that I've known has always had some kind of time-to-market reduction program. And so I don't know, when we hit zero, what will happen, but it's an ongoing thing. Everyone has it all the time.

And so what they've done is to say, "But the way we go about this is going to be different than what we've seen in some other settings. We don't tell people that they have to be twice as fast or twice as productive and let them figure out how to redesign their processes. We will focus on things that they can influence. We will focus on the ability to solve problems, to front-load problem-solving. And then, if that works, then good things will happen. Then we can cut development times, we can be more productive, and so forth.

And they literally did this. I mean, they measured this. So what you see here is the cumulative number of problems solved, percentage problems solved. Development—these are the different stages of development; so stages. And here are the different initiatives,

management and organizational initiatives that they had in probably the last twenty years. So what they did here is at the beginning. *S5* here is "start of pilot runs."

At this point, they had about half of all problems solved in a typical car development project. And in a typical car development project, you're dealing with tens of thousands of problems that you typically work on. And then they improved communication. They did joint problem-solving. They introduced three-dimensional computer-aided design. They introduced computer-aided engineering, which is essentially simulation. And all the time, they tried to measure whether their ability to solve problems earlier in development actually improves.

And they found that, over time, this curve actually shifted to the left, which was what they wanted. Because now when you look at this point, *S5*, almost all the problems have been solved, and they could start actually cutting out two or three of these phases at the end because they would really serve no additional purpose. And that's how they increased productivity.

So they didn't do it the other way around. They didn't say, "OK, let's cut this out, and then people will have to panic and figure out how to solve all their problems," which, of course, would lead to bad quality out in the marketplace. Huge risk for a company where quality is premium. They said, "No, no, let's first make sure that people know how to do all this, and solve problems early. And then we start cutting it down. So we don't face a quality risk in the marketplace.

And there are other companies who have followed the other approach, and it showed in their products when they came out. So this was a way to play it safe, and it also empowered people to do something, even at the lowest level. So the entry-level engineer could affect the ability, could affect problem-solving. May not able to affect large things like cutting development time in a team of 700 people down by 50 percent, but for that little area that he or she is responsible for, they could actually make some changes.

So, very, very powerful principle, very powerful introduction. And I want to basically close it here now.

Shifting the Locus of Experimentation

There are some final things shifting to a locus of experimentation. Just to make it appealing to you, this is the idea—and we've seen this model used very successfully in the semiconductor industry, now other industries as well, such as food and flavor industries—where you're using these tools to shift product development and innovation out to the customers, where the customers then essentially start to develop products on their own, and you become the manufacturer of that. And you're responsible for the tools development. In a semiconductor industry, this approach, if you track it back, created more than $10 to $20 billion dollars of value in the industry just at the component level. Very, very interesting concept, and it really is about migrating value out to customers as opposed to just keeping it internally in the company, which we focused on today.

Summary

And so I want to close it here, and maybe offer you a challenge. As you go away here, think about yourself and your own organizations. If I asked you right now, could you make a list of ten to fifteen experiments, organizational experiments that you're currently running?

If the answer is no, then the question is how will you come up with innovative products, services, or businesses if you're not experimenting? So I would go back and really challenge yourself in terms of maybe we should fund some of these things. And if the answer is yes, then the test is, be honest. Are they really experiments, or did they become experiments after the fact? That is, are they funded, designed, managed as an experiment, with learning objectives, clear questions at the beginning? Or as I said, do they just become experiments because they don't end up working, and therefore, they must have been experiments.

So that's what I want to leave you with at the end. Thank you.

