5

Solution: Prototype the Minimum "Awesome" Product

If you can increase the number of experiments you try from 100 to 1,000, you dramatically increase the innovations you produce.

—Jeff Bezos, CEO, Amazon.com

IN CHAPTER 4 WE describe how Banco Davivienda (BD) tried—and failed—to reach the large "unbanked" population in Colombia by creating simple bank accounts with lower fees. After its initial, solution-first approach failed, the bank sent teams into poor neighborhoods to understand the problem through

ethnography. This fly-on-the-wall approach gave managers a deep understanding of the job-to-be-done and a vision of what the solution would have to do for customers. But how did they actually develop a solution?

To get outside the box, the BD team tried some unusual things. First, the team members listed the key elements of a bank account, such as branches, accounts, debit cards, forms, signatures, paper, fees, and so on. Then they subtracted one item at a time and asked, "Can we build a business around this one item?"[1] Initially most team members scoffed. How could you build a solution using only one element? But when they suspended their disbelief, they realized that, for the unbanked, a simple solution to a specific problem might be the best approach. During their field research, teams noted that virtually everyone, including the unbanked, had cell phones. So the team studied wireless service providers and found that some companies had developed products that the unbanked would pay for, such as virally adopted games or a joke of the day. The team also began looking at other industries, such as internet retailers, to see how they provided solutions through mobile phones to a lower-income customer segment. Lastly, the team observed financial institutions in Asia, Africa, and other areas of Latin America to see how others were solving similar problems.

As mentioned earlier, eventually the BD team tested a mobile phone wallet that customers could use to do one thing: receive and make payments. The new solution stripped away traditional elements such as application forms, debit cards, and so on. Customers didn't have to come into a branch to sign documents or present personal ID; everything could be done on the mobile platform. To overcome compliance and regulation issues, Banco Davivienda borrowed emerging big-data techniques to analyze typical behaviors for similar customer groups and monitor or reject atypical usage.

When the team hit a substantial roadblock—how to let customers withdraw cash without adding an expensive and cumbersome ATM card program—the bank tried something highly unusual in banking but common among internet retailers: a one-time password, which, in this case, let customers withdraw cash from any ATM. Banco Davivienda's solution produced significant growth, with BD quickly enrolling more than a million users in various countries.[2]

Solution-Storming to Generate Solution Options

To develop its novel solution, the team at Banco Davivienda used a tool we call *solution-storming* (brainstorming multiple solution options) to help it search broadly for solutions before using customer tests to help it narrow the options to a single solution. Going broad in the search for a solution is a foundational principle for solution-storming—and for innovation generally. It leads to more options and combinations, and that in turn leads to novel solutions. For example, most people credit Henry Ford with developing the modern production line—an invention that allowed him to make the automobile affordable. Ford radically transformed the auto industry and American life. How did he do it?

We see the production line as a "solution," but Ford actually developed it by searching broadly, borrowing ideas such as interchangeable parts (used in sewing machines, firearms, and watches); continuous-flow manufacturing (used in processing flour, canning food, and making cigarettes); and assembly-line techniques (used in meat-packing plants and breweries).[3] In similar fashion, the BD team generated solution options by first searching broadly, in part by observing practices in other industries and other countries.

Starting Solution-Storming

Start your solution-storm with a problem and customer vision statement like the one described in chapter 4. As with all kinds of brainstorming, a key principle of successful solution-storming is that you don't shut down any proposed solution or solution process too early.

Recall that Banco Davivienda was skeptical that subtracting the elements of a familiar solution would work. We can top that: we once had a participant in a solution-storm for Leatherman, a manufacturer of multitools and pocket knives, propose shipping a live monkey with each tool to assemble the product. It took all we could muster to not shut down that idea immediately. But then the idea turned out to be pivotal in helping the team discover a super-simple assembly—one even a monkey could do easily.

To help you brainstorm solutions, we suggest you choose from a menu of techniques (see figure 5-1). You won't use all these techniques at once; instead, think of them as choices or options.

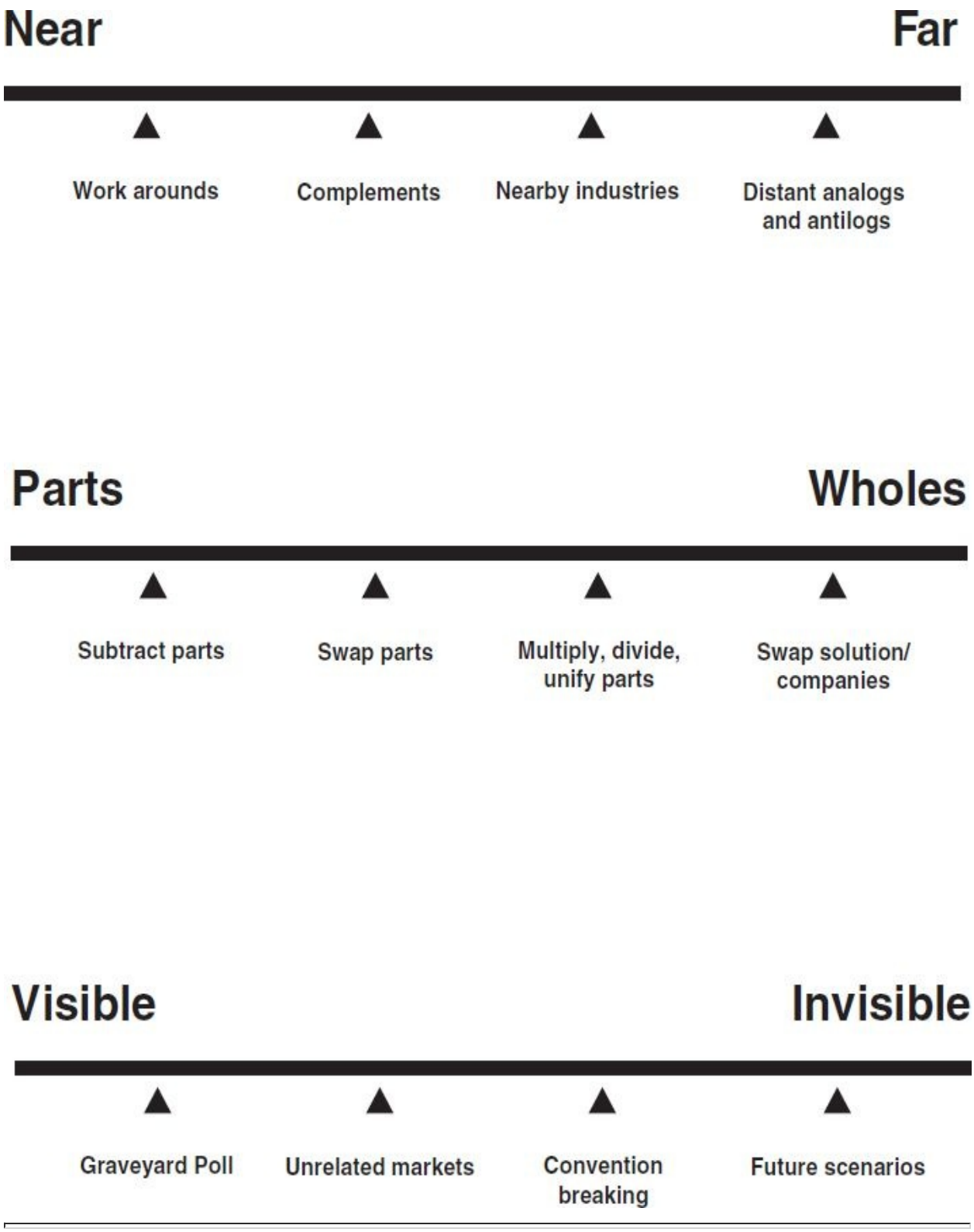
Analogs: Close and Far Away

Think about the possible solutions around you on a spectrum from those that are close to your industry to those that are far from your industry. Closest to you, check to see whether one of your customers has already developed a workaround. Usually, these make-do solutions are held together by figurative duct tape, but they provide insights into ways you could solve the problem. For example, one entrepreneur we interviewed initially felt discouraged when he discovered a potential customer had already developed a workaround to solve the problem. But then he licensed the solution for a small royalty and used it to build a solution and company that eventually reached a market value of well over a billion dollars.

Look for analogs and complements to your existing solutions that can suggest alternative solutions (be aware that there may be novel solutions in adjacent industries). For example, one fuel cell company we studied borrowed newspaper printing techniques to print fuel cell membranes. These examples of borrowing demonstrate the power of analogs of what to do or what not to do. For example, Rent the Runway borrowed analogs from Netflix and from airline reservation systems, and Banco Davivienda borrowed analogs from internet retailers and financial institutions in other emerging markets.

FIGURE 5-1

Tools for solution-storming


Near
Far
Work arounds
Complements
Nearby industries
Distant analogs and antilogs
Parts
Wholes
Subtract parts
Swap parts
Multiply, divide, unify parts
Swap solution/ companies
Visible
Invisible
Graveyard Poll
Unrelated markets
Convention breaking
Future scenarios

Elements: Parts and Wholes

Consider solutions in terms of parts and wholes. At one end of the spectrum, as with Banco Davivienda, subtract one element of an existing solution and try to build on it as the essential component. Or consider how you could swap in or swap out parts of a solution, as the iPod did by swapping in the movement of a combination lock to search rapidly for songs. Also think about how you could multiply, divide, or unify features. In the classic example, Gillette multiplied the blades in a razor to create a new solution. Or you can consider stealing the entire solution from another company by asking yourself questions like, “How would Amazon (or Apple, or Disney, or . . .) solve this?”

Observables: Visible and Invisible

Think about solutions that might be nearby but difficult to see. For example, others may have tried and failed to create the solution you seek. What can you learn from searching the graveyard of prior failures? Next, consider unrelated markets or disciplines to borrow an idea that could transform your industry. For example, ideas from biology (such as fitness landscapes or the process of variation, selection, and retention) have changed how we think about strategy and change. And look for ways to break the conventional wisdom of your industry. For example, IKEA broke the convention that furniture had to come assembled and thereby revolutionized its industry. Lastly, daydream about the future by ignoring the current technological limitations you see and imagine what the perfect solution might look like. For example, what would the perfect portable music device look like? The iPod is a great solution, but perhaps a better solution would be to allow customers to speak the song they want to listen to, say “play,” and then hear it in stereo sound. How might you make something like that happen? Examining “awesome” new products is an activity that might inspire you to imagine novel solutions.

Selecting Solutions to Prototype

After you've generated a range of solution options, you're ready to select the most promising ideas to test with customers. Recall Intuit's experience from chapter 1: when the team voted on the "best" ideas, they tended to pick those that were easiest to understand and implement. Wendy Castleman, the innovation catalyst community leader at Intuit, comments:

> Project teams had truly mastered "going broad"; they were good at generating lots and lots of ideas . . . But teams did a bad job of picking the best ideas to work on because our selection criteria were generally faulty. The problem stemmed from the team voting on ideas, a classic design thinking approach . . . The ideas with the most votes get explored further. But it turns out that people often vote for what is easy to implement and familiar, and that rarely yields ideas that will surprise and delight customers.[4]

Based on our observations at Intuit and other companies, we recommend you post the solution options on the wall and use the following process. First, think of the different themes or characteristics represented among the solutions, such as "ease of use" or "high performance." Then select an option from each theme to explore so that you can observe how customers respond.

Second, define the proposed solutions along a dimension or spectrum. For example, one spectrum might involve ranking solutions from low to high on dimensions such as "game changing," "bold/risky," "most easily attainable," "technical difficulty to deliver." Then select solutions to test at different ends of the spectrum. This approach will help you to go broad in the solutions you test.

Third, once you've selected solutions to test, write down your leap-of-faith assumptions or list them as questions to be answered. Then prioritize the assumptions based on those that you think are most important to validate in order for a solution to succeed. For example, when the Banco Davivienda team was brainstorming solutions for the unbanked, they wanted to test whether customers would sign up for bank accounts through their smart phones without visiting a branch or speaking to a person. It was critical to test this assumption for target customers, who didn't have the time or means to get to a branch. Once the critical assumptions have been made explicit, the team can design experiments (use prototypes) to test them.

Four Kinds of Prototypes

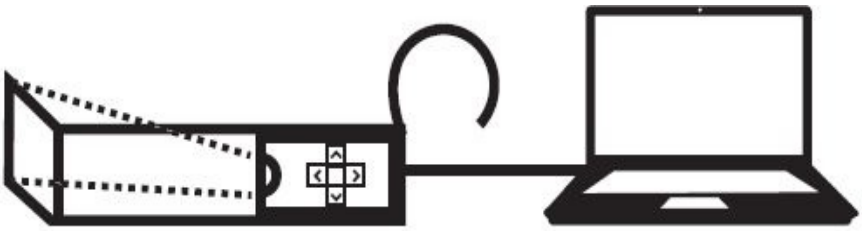
In a semisecret location, the Google X lab has been exploring crazy new technologies such as flying wind turbines and wifi balloons. These are the kinds of technologies, given the level of technical and demand risk, that most companies don't even consider. The first product, Google Glass, began to roll out in early 2012, with customers lining up to pay $1,500 to try a beta version. Although the success of Google Glass is anything but assured, the product development process offers lessons on prototypes.

The Google Glass team originated with a project by University of Washington professor Babak Parviz.[5] Although initially an interesting technology, it became an interesting solution when the team noticed a problem: How often in social interactions people "check out" to check their smart phones. The Google X team asked, "What if you could use this technology to stay engaged with the world around you while also using the internet?" At the same time, you can imagine the immensity of the technical challenges: How could you allow someone to connect to the internet using a lightweight, touchless device? Given the technical risk, ask yourself, How long should it take the team to create the first fully operational, wearable prototype that projects live images from the internet? Ready? It took one day.

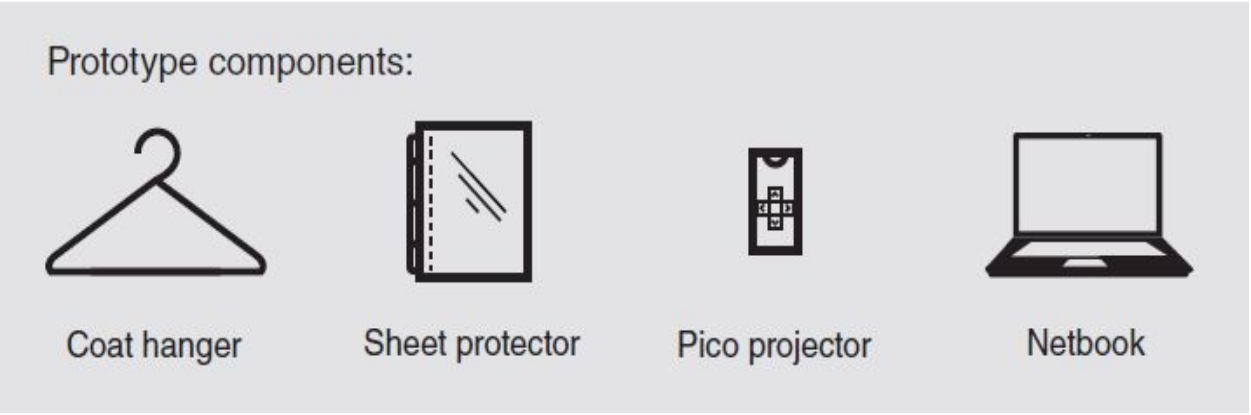
In a recent presentation, Tom Chi, head of experience at Google X, described how the team created and tested the first prototype. How did they do it? Check out the image in figure 5-2.

FIGURE 5-2

Google Glass first prototype



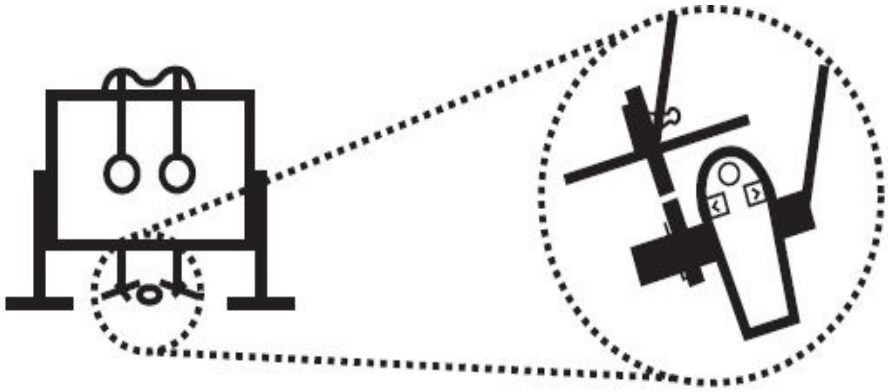
First prototype



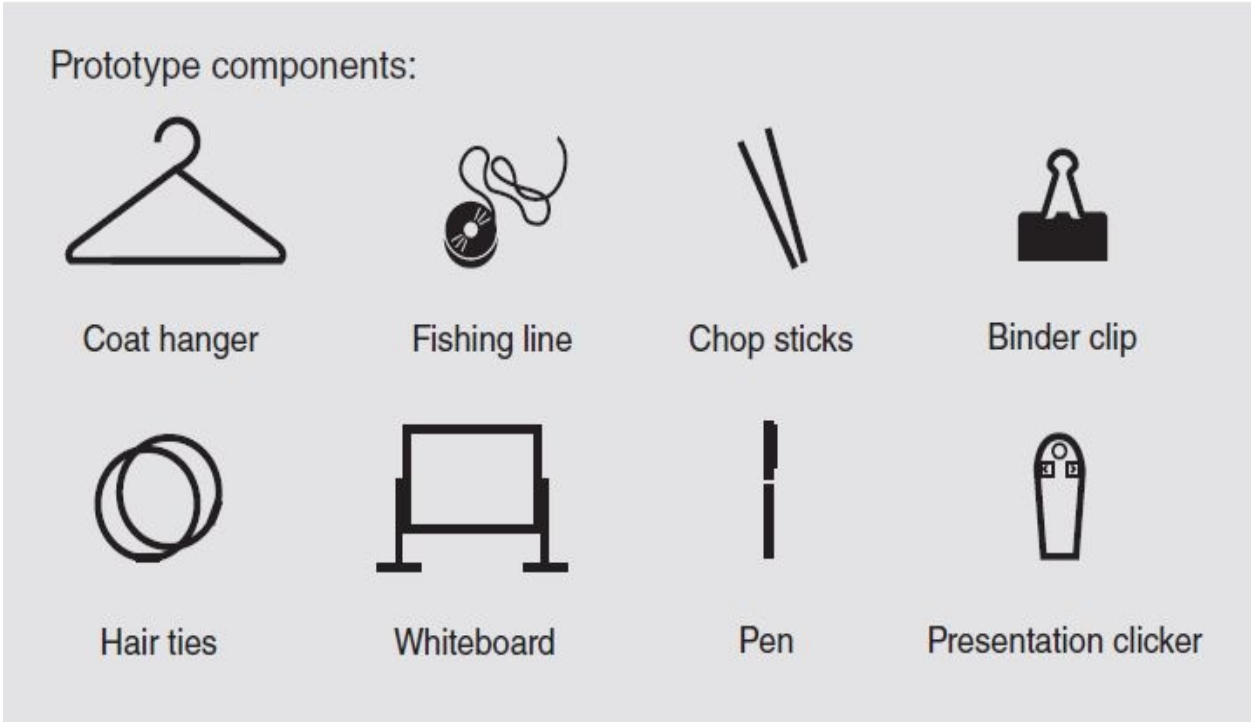
After creating a wearable device, Chi and the team needed a way to navigate it. If you've seen the movie *Minority Report*, you probably remember Tom Cruise moving his hands in the air to manipulate the computer. The Google X team also saw the movie and asked, "Why not try it?" Any guesses as to how long it took to prototype the motion detection system? Ready? About forty-five minutes. Chi and the team allowed users to manipulate the device by attaching borrowed headbands (worn around the wrist) to a clicking device (made from a pencil, a binder clip, a chopstick, and a mouse) via a taut fishing line (run over the back of a whiteboard) so that any movement by the user put tension on the line and clicked the device (see figure 5-3).

FIGURE 5-3

Google Glass navigation prototype



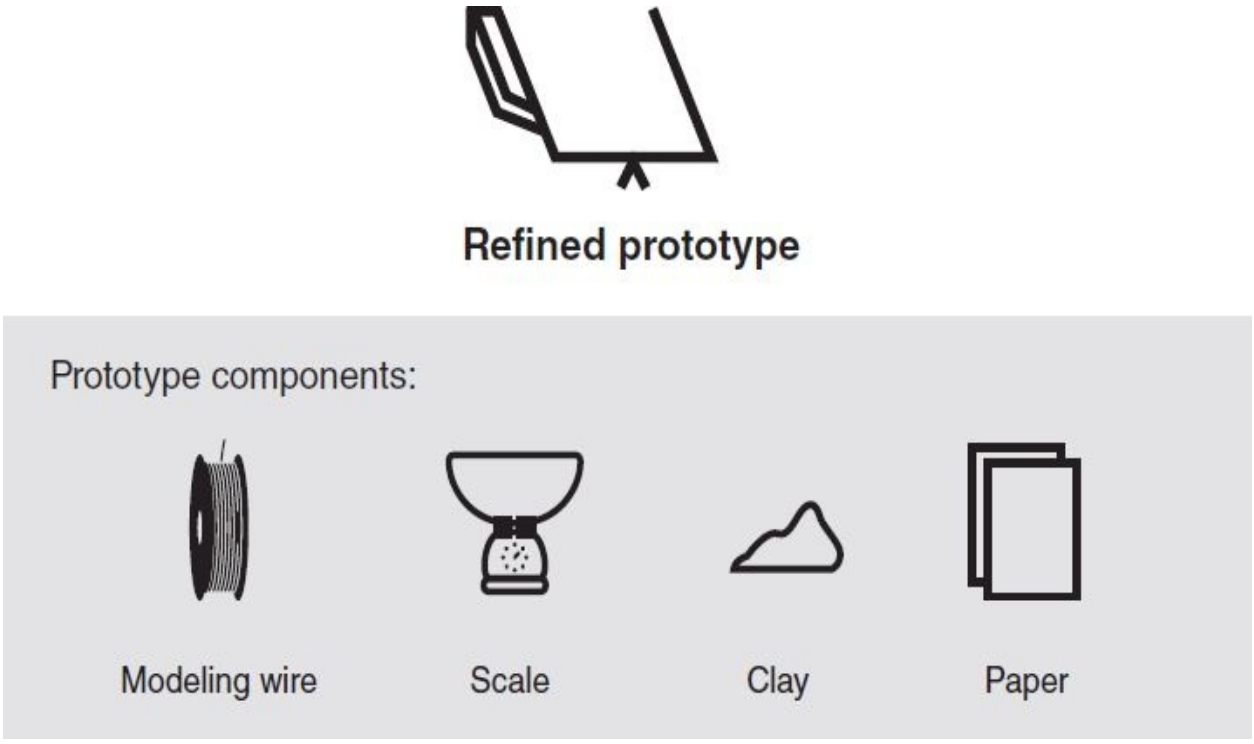
Second prototype



Right away the team learned that even though it worked for Tom Cruise, it looked and felt strange in practice, so they experimented with other approaches. Despite what at first appeared to be a series of dead ends, each rapid prototype helped the team nail down the final navigation system. And what about the glasses themselves? Although it easily could be overlooked, people hate to wear heavy glasses that press down on their nose. So what should you do about the weight? Chi and his team started prototyping using clay to weight different parts of the glasses and discovered that putting the weight behind the ears was quite tolerable (see figure 5-4).[6]

FIGURE 5-4

Prototyping Google Glass weight distribution



The Google Glass experience shows how quickly a company can move in developing prototypes, whether hardware, software, or services. And Google Glass is not an outlier. Gmail and AdSense (Google's content-targeted advertisement product that has made billions) were each prototyped in a day.[7] Indeed, we have found that at companies like Google, Intuit, and Valve, it is typical to build a prototype in twenty-four to forty-eight hours.

Although many people have heard about prototypes, few people understand the various types or know how to use them properly. Managers often go wrong by forgetting that every prototype should answer a specific question or by putting more effort into the prototype than is justified, simply because building stuff feels like progress.

Based on our research and practice, we recommend four types of prototypes. Here they are, in order from the simplest—with the fastest learning cycle—to the most elaborate, and thus slower learning cycles:

most elaborate, and thus slower learning cycles.

- Theoretical prototype
- Virtual prototype
- Minimum viable prototype
- Minimum awesome product

Theoretical Prototypes

The number 1 thing you can do to avoid the trap of building products customers don't want is to not build anything. We're not advocating inaction. But we are advocating a theoretical prototype as a tool to help you survive the early days of innovation ambiguity without falling into the *Field of Dreams* trap (the "build it and they will come" myth portrayed in the film). A *theoretical prototype* expresses your idea as a well-structured mental image in which you outline the general shape of the solution, but not the specifics. An example of a theoretical prototype might be to ask customers, "If you could check on your house visually from your smart phone, would you want to?"

The beauty of a theoretical prototype is that it's fast and cheap. You can test dozens within a week. For example, AT&T recently held a series of workshops with the members of its leadership team to build their innovation skills and consider ways to improve customer service. The attendees took fifteen minutes to solution-storm pilots or experiments to test new ways of delivering the industry's highest-quality customer experience. These were simple theoretical prototypes, such as testing a way for customers to walk into an AT&T store and make purchases without needing to interact with a human. These sessions led to the testing of a number of virtual and minimum viable prototypes during the next twelve months.

By contrast we've watched many teams dive into building products based on their intuition alone before they've even tested whether anyone cared about the solution. In fact, the biggest problem for most corporations is that when they do a prototype, they do it in a complicated, expensive way, using lots of 3-D modeling and other techniques that are inappropriate at such an early stage. A theoretical prototype gives you a cheap and easy way to "test" and adapt solution concepts with customers. Generally when you talk to customers about the jobs to be done, you will use a theoretical prototype (or sometimes a hyper rapid virtual prototype) as a straw man for the conversation. At the point when you're getting enthusiastic, positive responses to your theoretical prototype from potential customers, you're ready to create a virtual prototype.

Virtual Prototypes

Several decades ago, IBM wanted to test an unusual solution to a common customer problem: transforming spoken words into type. A team of computer scientists and executives envisioned highly sophisticated voice recognition software that would transform dictation into text. In addition to the core software, the solution would require high-quality microphones and an adaptive algorithm to parse differences in dialect and diction. Although the technical risk seemed high, the team believed that if it could overcome the technical challenges, it would solve a huge customer problem.

But before it made a big investment, how could IBM test whether customers cared about the solution? At a minimum, it seems that some primitive beta software would be needed to test whether the solution would be worth building. Instead, the IBM team hung a sheet across a room, and, when the customer spoke, a hidden typist on the other side of the sheet captured what was said and projected it on a computer screen for the customer to see. Essentially IBM "pretended" to have the solution to answer the question, "Do customers care?"[8] Like IBM and the Google Glass team, most successful managers use *virtual prototypes* (VPs)—"pretend-otypes"—to answer key questions while avoiding the costs of developing expensive, unwanted solutions.

To develop a VP, ask yourself a simple question: If I had to sell the solution today, how could I fake it in a way that feels realistic? The answer should be a good guide for what you could develop and how fast. Our favorite VPs use PowerPoint, sketches, off-the-shelf components, or other mock-up tools. For example, PowerPoint can mimic software by creating invisible click-through hot spots to quickly simulate what might take months in development. Similarly, Kaiser Permanente faked various service innovations through a combination of storyboards and rehearsed service simulations. One observer said, "There's something magical about low-fidelity ways of trying something out. It automatically allows people to feel like they can put their fingerprint on it. The more polished, the more people feel like it's already done."[9]

Whatever tool you use—including advanced tools like 3-D printers, video, or flash demos—remember that all prototypes should be designed to answer specific questions. Your VP will likely be an imperfect representation of the product, but the goal is to start answering your key questions, or hypotheses, about what customers want. By doing only the minimum to get feedback on the most relevant uncertainty you face, you speed up your learning and preserve

your flexibility. The ability to learn quickly through prototypes (especially VPs) may be your most important advantage over slower rivals. In the words of one innovator, “A chess novice can defeat a master if moving twice each round.”[10]

Minimum Viable Prototypes

When someone at social gaming giant Zynga has an idea for a new game, no one starts the process by building the game. Instead, the team members boil the idea down to five words and perform a smoke test, as described in chapter 4, to see whether there is any customer interest. For example, suppose someone has an idea for a game about running a hospital. She would put up a low-cost ad on a high-traffic web page that simply says, "Ever fantasize about running a hospital?" The users who click on the link receive a brief description of the theoretical prototype and are told they will receive an e-mail when development is complete.

If the ad produces enough response, Zynga designers then spend a week or less building a minimal, stripped-down version and launch it to the e-mail list. Then they see what they can learn about the solution: How many users sign up, how long they play, what features they like, what they hate, and so on.[11] This one-week test, or what has been called a *minimum viable prototype* (MVP), helps them learn crucial lessons about what customers actually want and guides them in deciding whether to perform another iteration or go explore different ideas.

A minimum viable product has been defined as a product having the minimum feature set required to work as a stand-alone product while still solving a "core" problem (we use the word *prototype* rather than *product* as used by Eric Ries in *The Lean Startup*, to emphasize that your objective is to test your assumptions rather than build a product).[12] You build an MVP to rapidly test which features are most likely to drive customer purchases. It's like an exercise in archery: your goal is to hit the bull's-eye features that drive purchases and put other features on the back burner. Why? It's easy to be swamped with feature ideas that could add value. Adding features feels as if it will increase the chances customers will like the product, but actually it can decrease customers' ability to recognize how your solution solves *their* problem (see "Don't Forget to Go Narrow"). Worse, you waste time building features that don't motivate customers to purchase. You may not fully identify the bull's-eye in the beginning, so use the VPs and MVPs to identify those features that matter most to customers. At first, the process may feel haphazard and random. But remember that you can use many MVPs to test minimal feature sets, as quickly as you can, to get multiple points of useful feedback from target customers.

Don’t Forget to Go Narrow

When the cofounders of an educational software company first met with the chief technology officer at the University of Southern California, his feedback on their beta product seemed fairly positive. Earlier conversations with other faculty and university officers had established a significant problem with existing learning management software. But to get a sense of how well they were solving the problem, members of the team showed the CTO their minimum viable prototype: essentially an inexpensive flash-based version of the software that highlighted the top twenty features they believed would be most valuable for educators. Then they asked a critical question: “How much would you be willing to pay for a solution like this?” Despite the CTO’s early enthusiasm, his response—$2,500 per year—dimmed their hopes of building a sustainable business.

But before leaving the room, they tried a tool we recommend called the $100 R&D game.[13] You ask customers to allocate $100 of an R&D budget to the features they most want. When the developers asked this question, they were surprised that the CTO allocated $80 to a simple drag-and-drop feature and then split the remaining $20 between two other features, a pattern repeated in visits to other universities.

Using this data, the team crafted a new flash prototype and returned to the first CTO, this time with a minimum viable prototype that contained only three features, the most prominent being the drag-and-drop feature. What does your intuition tell you about how the CTO responded to the stripped-down prototype? In fact the CTO liked the product more and offered to license the product for $12,500 per year. By focusing on the minimum feature set, the team sold an early product with only one-tenth the features for five times the original expected price.[14] Throughout the book we emphasize the importance of diverging and going broad first, but as you develop MVPs, don’t forget to converge and go narrow to really nail the solution.

In addition, rather than add many features to a single prototype, it’s a good idea to develop several prototypes that emphasize different key features, and see how customers react (see “Won’t an MVP Hurt My Brand?”). Even testing a solution missing the key feature has power. Remember, the goal is to learn, and

it’s easier to learn when you’re testing fewer dimensions in a single MVP. One team we worked with tested a medical device lacking the key feature they felt added value (which was also slowing FDA approval) and discovered an entirely new market that didn’t actually want that “critical” feature.

Won’t an MVP Hurt My Brand?

Some folks are concerned that offering a low-fidelity MVP to the market might damage the company’s brand. For products in mature categories where customers have a developed understanding of the product and clear expectations (conditions of lower uncertainty), it’s true that MVPs can be dangerous because customers will demand what Geoffrey Moore calls “the whole product solution.”[15] But when companies launch new products that are not well understood, they’re often adopted first by innovators who have lower expectations and are more forgiving of an MVP. Therefore, when you test your MVP with a sample of customers, you can get away with a minimal prototype. Of course, you will be embarrassed about it. In fact, as Eric Ries argues, if you’re not embarrassed, you’ve done too much work. You didn’t develop an MVP.

To help allay this concern, we recommend several tactics. First, create a separate brand or sub-brand that clearly communicates the preliminary status of the product. For example, Intuit launches all MVPs with the “Intuit Labs” brand, and Google uses “Google Labs.”

Second, flip the liability of testing an MVP on its head by providing higher levels of service and satisfaction for the MVP test. For example, when Samsung launched an experimental refrigerator in Southern California, it offered customers a hotline with white-glove service (the company would replace the fridge and all its contents within twenty-four hours if something went wrong). An acquaintance actually participated in this experiment, and his fridge had a problem. He told us, “They responded so well to my problem that I’m only going to buy Samsung products from here on out.”[16]

And remember, you are testing the MVP with a sample of customers and not the entire population. You may think of a sample in terms of limited geography (e.g., a particular town), type of customer (e.g., early adopters), or a limited number of customers.

Minimum Awesome Products

Once you have a minimum viable prototype and have validated your core assumptions, your next step is to develop something that customers cannot resist, something that customers love, something awesome. We first heard the term *minimum awesome product* (MAP) while spending an afternoon observing Intuit's globally broadcast training session for all the product designers, developers, and user-interface architects in the company. The training was focused on a single issue: What is "awesome"?

Intuit had presented lean start-up training, and everyone was familiar with the concept of a minimum viable product. However, founder Scott Cook was uncomfortable with some aspects of the concept. "When you say 'minimum viable product,' engineers naturally focus on the word *product.* So they want to jump to building a product," says Cook. "At the early stages of a new product idea, we want our engineers to just be experimenting, answering their leap-of-faith hypotheses."[17] We agree with Cook, and that's why we prefer to substitute the word *prototype* for *product* in our use of the term *minimum viable prototype*. But once the MVP has revealed which features are most likely to drive a customer purchase, it's time to go beyond viable and reach for awesome.

The goal of a minimum awesome product is to deliver a solution that is so extraordinary on the most important dimension that it inspires positive emotion in your customers. As Scott Cook explains, "You don't want to be *viable* in the dimension that matters—you want to be *awesome* in the dimension that matters, all while maintaining an uncomfortably narrow focus." In other words, you want to identify the minimum feature set possible and then relentlessly focus on making your solution awesome on those dimensions. But what is awesome? And how do you get there?

Customers describe products as awesome when they inspire positive emotions, such as creating deep satisfaction, calming anxiety, or giving confidence. Often "awesome" solutions do unexpected things that inspire positive emotion. When a product or service surprises you by doing something you didn't expect—something you may not have even thought was possible—it can evoke positive emotion and prompt you to say, "That is awesome!"

Apple has often achieved awesomeness in this way. When the first iPod launched, customers said, "Wow! I can really carry my whole library of songs with me everywhere I go. That is awesome." When the first iPhone launched, many customers said, "Wow! I can do this (listen to music, take a photo, find an

address, etc.) on my phone. That's cool." Steve Jobs was known to say that Apple's job was not to give customers what they wanted. It was to give them what they didn't know they wanted (or needed). That's when you evoke positive emotion.

Of course, that's a high hurdle. What if the product just nails the job-to-be-done perfectly but doesn't do it in an unexpected way? For example, Dyson vacuums are touted as having "twice the suction of any other vacuum." Can better suction really create positive emotion? The answer: absolutely. But the solution must be noticeably better than alternative solutions (and of course, the customer must care about suction). The solution evokes positive emotions because the customer is surprised by how much better the offering is than competitive offerings.

To illustrate, Intuit identified what it thought was a problem for a large group of Americans: simple tax filers struggling to fill out their complicated tax forms. One thing Intuit learned from early experiments—using theoretical and virtual prototypes—was that simple filers often had trouble figuring out which information from their W-2 statement to plug in to their tax form. (For tips on getting the most from your experiments, see "How to Run a Good Experiment.") So the Intuit designers tested an MVP that let filers take a photo of their W-2 form with their cell phone or camera and upload it to their computer; then the software pulled the data from the W-2 into the right location on the tax form. Tests showed that filers loved the idea of taking the photo and having the data magically appear at the right location. But they didn't like the ensuing steps of uploading it to their computer, because it was both time-consuming and complex. "Why can't I just take the picture of the W-2 and finish my taxes on my phone?" many asked.

Intuit listened, creating a prototype that took the W-2 photo, plugged the data into a tax app, and quickly completed the taxes (after asking the user a few basic questions) on the smart phone. This dramatically cut the time and complexity of tax preparation for the simple filer. Initial tests revealed that the app, dubbed SnapTax, let simple filers take the photo and finish and file their taxes in less than ten minutes (something Intuit touted in ads). That was unexpected. And it evoked strong positive emotional responses from users. One early user of SnapTax gave it a five-star rating, gushing that it was so easy to use he literally completed his taxes during a Valentine's Day date. The couple were so happy to be finished with their taxes they had one of their best dates ever. Can you imagine a tax preparation product inspiring that kind of a response from a user?

To develop a minimum awesome product, Intuit focused on simple filers (some 59 million people), identified the key pain points, and then persisted in searching for a solution to that pain in an unexpectedly easy way. Of course, minimum awesome products start out as satisfactory MVPs. In fact, with the first smart phone version of SnapTax, more than 90 percent of users abandoned the app after three touches on the phone. Why? It was because "the first screens required people to create an account with name, password, et cetera," says SnapTax product development head Amir Eftekhari. "But users just wanted to check out the app to see how it worked. They didn't want to create an account."[18] So Intuit redesigned the app so that within three touches, users could see the three steps to complete their taxes and capture a photo of their W-2 to start the process. Account setup, and payment, wasn't necessary until the user was ready to file—but that was typically within ten minutes. The point is that you use the MVPs to quickly improve those dimensions of your solution that can inspire emotion—and turn your MVP into an MAP—a Minimum Awesome Product.

As you refine your prototype, remember that your customers are hiring your solution to do a job for them—and that every job has functional, social, and emotional dimensions. Your solution may inspire awesome by doing the unexpected on any of these three dimensions of the job. But beware: it can be difficult to be awesome to everyone. Stay "uncomfortably narrow" as you build your minimum awesome product.

How to Run a Good Experiment

Most people believe they understand experimenting, but when we teach executives and students, 90 percent of the time they get it wrong at first. What are the mistakes they make?

- **No hypothesis or metrics**. Don't conduct an experiment without a clear hypothesis or without defining how you will measure what you learn.
- **Wrong tools**. Don't rely on focus groups. They tend to get stuck in groupthink. And avoid starting with surveys. You don't yet know what questions to ask.
- **Bad samples**. Make sure you talk to a relevant sample of customers. Often experimenters pick the wrong people (for convenience), or

they talk to everyone (the population). Also, don’t reformulate your solution between every conversation; wait for a pattern to emerge, and then reformulate.

In contrast, good experiments test the most important assumptions quickly, reliably, and affordably on a relevant customer sample. And they measure the results. Consider the following simple experiment at Intuit. The finance operations team wanted to retain small-business subscription customers whose credit cards were about to expire. It had been sending e-mails to these customers, but a large percentage didn’t respond and when credit cards lapsed, the income stream from a customer was typically lost. So the team, looking for a better way, wanted to verify that customers were receiving the e-mails and whether the e-mails were effective at triggering a response. “The idea was to call customers to see if they had received our communication and get them to take action immediately, instead of putting it off—small-business people are busy folks who forget about things that aren’t urgent,” said Wendy Castleman, an Intuit innovation catalyst. So the team ran an experiment to test the hypothesis that many customers weren’t getting Intuit’s e-mail. When team members called a dozen customers selected because they represented different customer segments, they discovered that about a quarter had an outdated e-mail address and weren’t getting the e-mails. Others didn’t feel the urgency with an email communication. The team then tried another experiment: they hired a small team of temp workers to call businesses as their credit cards came due for renewal. The new communication process resulted in more than $8 million in recovered revenues in FY2013.

This experiment quickly tested a clear hypothesis, using customer behavior as the data, with clear metrics of success. Although this experiment worked out, Intuit has conducted hundreds of experiments in which the team quickly learned what doesn’t work and why, and pivoted to explore other ideas.

Have You Nailed the Solution? Three Tests

Using virtual prototypes and MVPs can be an illuminating but ambiguous process. How do you know if you have it “right,” the kind of solution where people say, “Wow, they really nailed it”? There are a few tests we recommend that you can use to “nail it, then scale it” rather than the other way around.

The Wow Test

If you're in the early days of theoretical or virtual prototypes, we recommend the *wow test* to measure how excited customers are about your solution. The wow test has two parts. First, qualitatively, when you show customers your prototype, can you see their enthusiasm, or are they only being polite? If you can't see the enthusiasm in their faces or hear it in their voices, then you probably need to make a change.

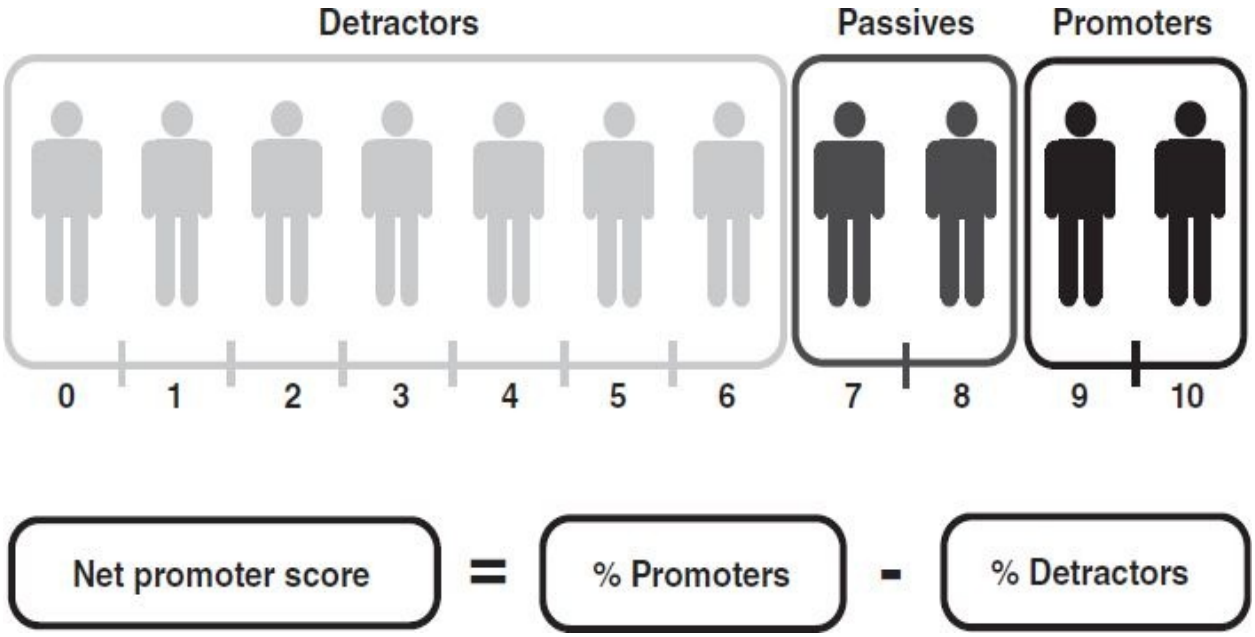
Second, quantitatively, ask customers, on a scale of 1 to 10, how excited would you feel to own the solution (10 being "extremely excited" and they're ready to buy the solution, and 1 being not at all excited)? You should be aiming to improve the wow score over time, shooting for an average greater than 7. Anything significantly lower than 7 may suggest customers are only being polite and you need rethink your solution.[19]

The NPS Test

Once you have an MVP or MAP that customers can start using, you can apply the promoter test. There are a few variants on this test, our favorite being the net promoter score (NPS). Recall that NPS is based on a single question: how likely are you, on a scale of 1 (not at all likely) to 10 (extremely likely), to recommend this product or service to a colleague or friend? "Promoters" answer 9 or 10, "passives" answer 7 or 8, and "detractors" answer from 1 to 6. A company's (or product's) NPS is the percentage of promoters minus the percentage of detractors (see figure 5-5).

FIGURE 5-5

Net promoter score



Your ultimate goal is to score 9 or higher with your core customer group. But you won't start there, so don't get discouraged. Instead, use the prototypes we've discussed to iterate there. When 80 percent or more of your core customers rate a solution 9 or 10 and your average NPS is higher than 60 percent, then you've not only nailed the solution but also created evangelists for your product. Notice,

too, that we said "core" customer group, and not all your customers: one of your greatest challenges is to stay narrowly focused on a customer group. Ten customers may ask you for dozens of features that can pull you in multiple directions. As mentioned earlier, if you try to please everyone, you will never nail the solution or turn your customers into evangelists. As you get feedback on your prototypes, look for the themes that segment customers into groups based on the functional, social, and emotional dimensions of the job-to-be-done, and then remove some customers from your focus (tell yourself you'll serve them later). For example, when Google engineer Paul Buchheit was building a prototype of Gmail he was told to find one hundred "happy" users. "I was like, 'Oh, that's easy, Google has like thousands of employees,'" said Buchheit.

> But it turns out that happiness is a really high bar, and to get people to say they're happy is actually sort of challenging. We literally did it one user at a time. We would go to people and ask, "Okay, what's it going to take to make you happy?" And in some cases they would ask for something really hard, and we'd be like, "Okay, well, you're not going to be happy with Gmail, quite possibly ever." But with other people it turned out there would be something really small we could do and then they'd be happy. So we'd do the really easy things until we got 100 people who were happy. And 100 doesn't sound like a lot but it turns out people are pretty similar to each other so if you can make 100 happy, you can usually make more [happy].[20]

So start by shooting for a 9 or 10 NPS score with ten people, and then you can think about progressing to the hundredth.[21]

The Payment Test

The ultimate test of whether you've nailed the solution is that people will pay you for it. Understandably, some managers may be afraid to ask the hard questions about whether customers (or end users) will pay. But delaying the question delays discovery of whether you have a viable solution. We've seen many instances when potential customers responded enthusiastically to prototypes, but when it came time to open their wallets, they lost their enthusiasm. In other cases, we've seen customers prepay for a product that doesn't yet exist. In one of our favorite examples, Coin (a single card device that stores multiple credit cards) presold millions of cards that weren't delivered until a year later. In truth, its story is nothing special: crowd-funding—which resulted in more than $5 billion in transactions in 2013—employs a form of the payment test in which funders prepay for an item.[22]

At the core, the payment test involves asking customers to pay for your solution, whether or not you actually collect the money. In most cases you conduct the payment test during the MVP stage or later. During the test, you need to get a credible commitment from customers. Just asking customers whether they will pay yields weak results. Actual behavior counts. For example, Intuit gives testers of its prototypes the opportunity to preorder or purchase using a credit card. After testers enter the first four numbers, a message pops up saying their contact information has been registered and they'll be contacted to purchase the product after it has passed final testing.

At this point, we should clarify a critical point: we are not asking you to go out and sell. Although that may sound contradictory, selling often reinforces a one-way communication pattern that shuts down your openness to feedback. Rather, when you conduct the payment test, keep in mind you're actually seeking customers' feedback. If your customer prepays, wonderful! But if customers hesitate or stall, then view that as a good outcome, too: it gives you the opportunity to directly ask customers what is still missing from the solution.

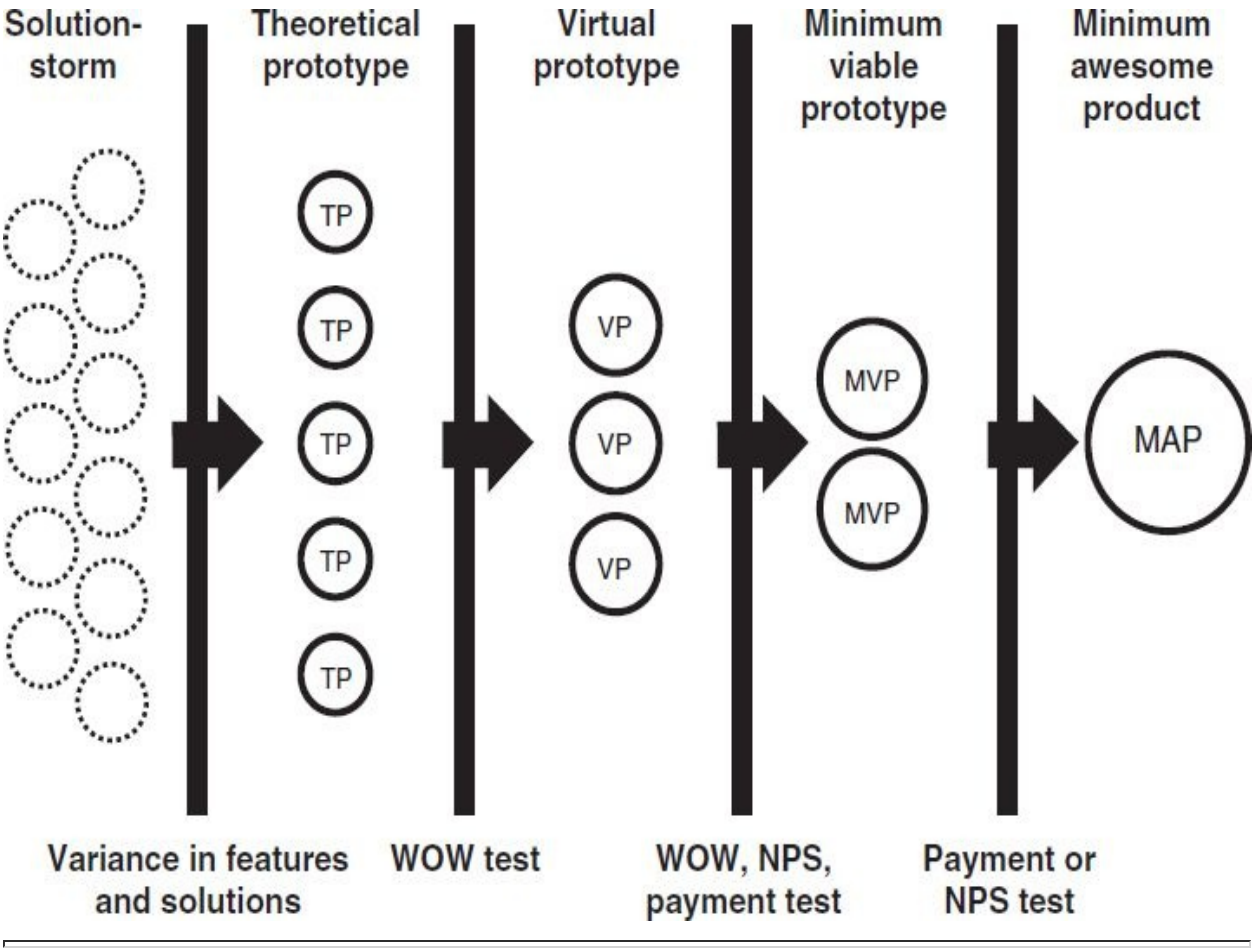
Don't wait too long to apply the payment test, because it's the ultimate test of whether your solution does the job. We've heard too many stories from managers who delayed the payment test, only to realize much later that their solution was neither viable nor awesome.

A Solution Process Template

To map the process you'll use to nail the solution, consider applying the solution process template shown in figure 5-6. The template reflects the approach we've suggested, starting with a solution-storm that generates a wide range of solutions. Then select several solution options to rapidly test with customers using theoretical prototypes. Then use the wow test to identify which solutions attract the most attention. Next, develop at least three virtual prototypes that embody different aspects of the solution, and test them with customers. Eventually you will converge to one virtual prototype using the tests described here, and you can develop it into a minimum viable prototype. Once you reach the MVP stage, you should use the NPS or payment test to determine which key features to highlight and perfect as you iterate your way to a minimum awesome product.

FIGURE 5-6

Solution process template



Watch Out: Don’t Get Trapped by Your Capabilities

Large companies develop core competencies that are key to their success. And executives are often encouraged to stick to their knitting—to apply the competencies that got the company to where it is now. But large companies that stick to their knitting face a big watch out. Although there are a handful of companies that have strayed from their core competencies and failed, there are mountains of cases in which companies, in times of uncertainty, stuck to their knitting and missed a huge opportunity or were killed by innovation.

Going beyond your current competencies is important for another reason: people typically develop new solutions by combining technologies and ideas from different fields. This means that a broad search typically outperforms a narrow search in generating solutions. While it feels more efficient to search narrowly within the company, it tends to result in incremental solutions that don’t delight customers

tends to result in incremental solutions that don't delight customers. Searching a broad range of technologies and firms (possibly through alliances) is critical for large companies if they hope to solve new problems and enter new markets. Amazon is one of the few large companies we know that has developed a broad range of technologies internally—and with other firms in partnership—that it draws on to generate solutions to help it enter new businesses.

Getting to Awesome

Some 90 percent of initial proposals don’t nail the solution to a significant problem. This explains why it’s folly to start by building a product or service before you’ve discovered how it falls short. It also explains why it’s important to know how to test many options by using prototypes. Adopting solution-storming and prototyping will minimize your investment, accelerate your learning, and allow you to test the assumptions that can kill your idea. We encourage you to use the cycle of searching broadly and then creating theoretical, virtual, and minimum viable prototypes, along with a minimum awesome product, to test your hypotheses and answer crucial questions. Remember to design your prototypes to answer specific questions and to use clear metrics to assess them. Embrace the freedom of using prototypes to validate your solution as the key to success under conditions of uncertainty.