11 Application: Toolkits for User Innovation and Custom Design

An improved understanding of the relative innovation capabilities of users and manufacturers can enable designs for more effective joint innovation processes. Toolkits for user innovation and custom design illustrate this possibility. In this new innovation process design, manufacturers actually *abandon* their efforts to understand users' needs accurately and in detail. Instead, they outsource only *need-related* innovation tasks to their users, who are equipped with appropriate toolkits. This process change differs from the lead user search processes discussed earlier in an interesting way. Lead user searchs identify existing innovations, but do nothing to change the conditions affecting user-innovators at the time a new product or service is being developed. Toolkits for users, in contrast, do change the conditions potential innovators face. By making innovation cheaper and quicker for users, they can increase the volume of user innovation. They also can channel innovative effort into directions supported by toolkits.

In this chapter, I first explore why toolkits are useful. Next, I describe how to create an appropriate setting for toolkits and how toolkits function in detail. Finally, I discuss the conditions under which toolkits are likely to be of most value.

Benefits from Toolkits

Toolkits for user innovation and design are integrated sets of product-design, prototyping, and design-testing tools intended for use by end users. The goal of a toolkit is to enable non-specialist users to design high-quality, producible custom products that exactly meet their needs. Toolkits often contain "user-friendly" features that guide users as they work. They are specific to a type of product or service and a specific production system. For

example, a toolkit provided to customers interested in designing their own, custom digital semiconductor chips is tailored precisely for that purpose—it cannot be used to design other types of products. Users apply a toolkit in conjunction with their rich understanding of their own needs to create a preliminary design, simulate or prototype it, evaluate its functioning in their own use environment, and then iteratively improve it until they are satisfied.

A variety of manufacturers have found it profitable to shift the tasks of custom product design to their customers along with appropriate toolkits for innovation. Results to date in the custom semiconductor field show development time cut by 2/3 or more for products of equivalent complexity and development costs cut significantly as well via the use of toolkits. In 2000, more than $15 billion worth of custom integrated circuits were sold that had been designed with the aid of toolkits—often by circuit users—and produced in the "silicon foundries" of custom semiconductor manufacturers such as LSI (Thomke and von Hippel 2002). International Flavors and Fragrances (IFF), a global supplier of specialty flavors to the food industry, has built a toolkit that enables its customers to modify flavors for themselves, which IFF then manufactures. In the materials field, GE provides customers with Web-based tools for designing better plastic products. In software, a number of consumer product companies provide toolkits that allow people to add custom-designed modules to their standard products. For example, Westwood Studios provides its customers with toolkits that enable them to design important elements of their own video games (Jeppesen 2005).

The primary function of toolkits for user design is to co-locate product-development and service-development tasks with the sticky information needed to execute them. Need-intensive tasks involved in developing a particular type of product or service are assigned to users, along with the tools needed to carry those tasks out. At the same time, solution-intensive tasks are assigned to manufacturers.

As was discussed in chapter 5, problem solving in general, and product and service development in particular, is carried out via repeated cycles of learning by trial and error. When each cycle of a trial-and-error process requires access to sticky information located at more than one site, co-location of problem-solving activity with sticky information is achieved by repeatedly shifting problem solving to the relevant sticky information sites as product development proceeds.

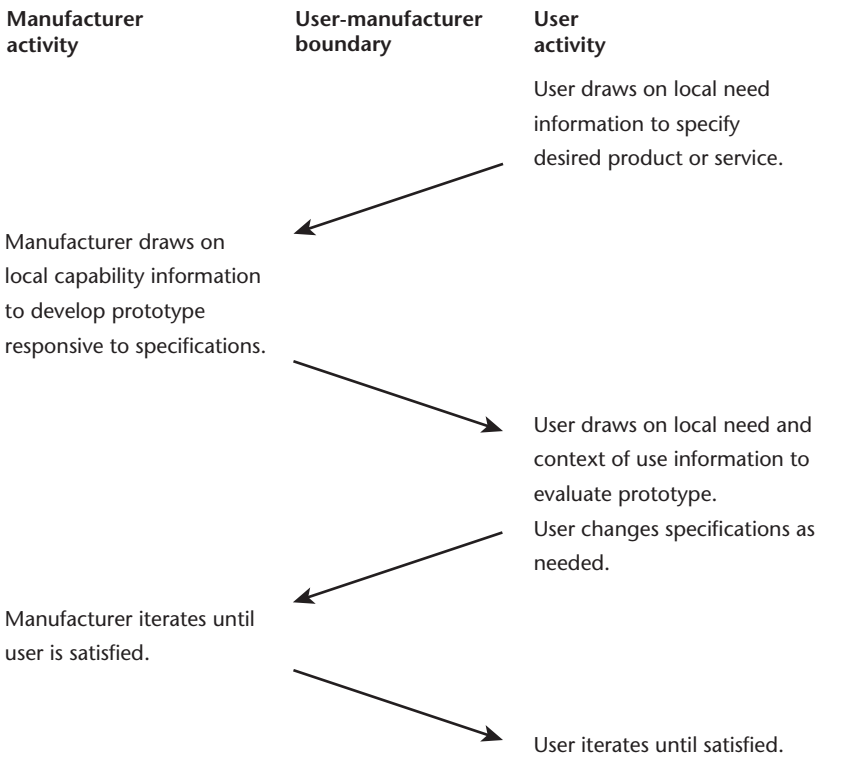


Figure 11.1
A pattern of problem solving often encountered in product and service development.

For example, suppose that need information is sticky at the site of the potential product user and that solution information is sticky at the site of the manufacturer. A user may initiate a development project by drawing on local user-need information to specify a desired new product or service (figure 11.1). This information is likely to be sticky at least in part. Therefore, the user, even when exerting best efforts, will supply only partial and partially correct need and use-context information to the manufacturer. The manufacturer then applies its solution information to the partially accurate user information and creates a prototype that it thinks is responsive to the need and sends it to the user for testing. If the prototype is not satisfactory (and it often is not), the product is returned to the manufacturer for refinement. Typically, as empirical studies show (Tyre and von Hippel 1997; Kristensen 1992), sites of sticky need and / or solution information are

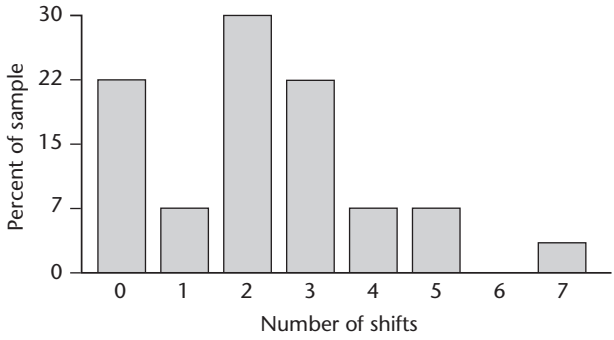


Figure 11.2
Shifts in the location of problem solving from user site to lab observed during process machine debugging. Source: Tyre and von Hippel 1993, figure 2.

repeatedly revisited as problem solvers strive to reach a satisfactory product design (figure 11.2).

Explicit management of user-manufacturer iterations has been built into a number of modern product-development processes. In the rapid application development method (Martin 1991), manufacturers learn to respond to initial user need inputs by quickly developing a partial prototype of a planned product containing the features likely to be most important to users. They deliver this to users, who apply it in their own setting to clarify their needs. Users then relay requests for changes or new features to the product developers, and this process is repeated until an acceptable fit between need and solution is found. Such iteration has been found to "better satisfy true user requirements and produce information and functionality that is more complete, more accurate, and more meaningful" (Connell and Shafer 1989).

Even with careful management, however, iterative shifts in problem solving between users and manufacturer-based developers involve significant coordination costs. For example, a manufacturer's development team may be assigned to other tasks while it waits for user feedback, and so will not be immediately able to resume work on a project when needed feedback is received. It would be much better still to eliminate the need for cross-boundary iteration between user and manufacturer sites during product development, and this is what toolkits for user design are intended to do. The basic idea behind toolkits for user design is, as was mentioned earlier, to partition

an overall product-development task into subproblems, each drawing on only one locus of sticky information. Then, each task is assigned to the party already having the sticky information needed to solve it. In this approach, both the user and the manufacturer still engage in iterative, trial-and-error problem solving to solve the problems assigned to them. But this iteration is internal to each party—no costly and time-consuming cross-boundary iteration between user and manufacturer is required (von Hippel 1998, 2001; Thomke and von Hippel 2002; von Hippel and Katz 2002).

To appreciate the major advantage in problem-solving speed and efficiency that concentrating problem solving within a single locus can create, consider a familiar example: the contrast between conducting financial strategy development with and without "user-operated" financial spreadsheet software:

- Before the development of easy-to-use financial spreadsheet programs such as Lotus 1-2-3 and Microsoft Excel, a firm's chief financial officer might have carried out a financial strategy development exercise as follows. First, the CFO would have asked an assistant to develop an analysis incorporating a list of assumptions. A few hours or days might elapse before the result was delivered. Then the CFO would use her rich understanding of the firm and its goals to study the analysis. She would typically almost immediately spot some implications of the patterns developed, and would then ask for additional analyses to explore these implications. The assistant would take the new instructions and go back to work while the CFO switched to another task. When the assistant returned, the cycle would repeat until a satisfactory outcome was found.
- After the development of financial spreadsheet programs, a CFO might begin an analysis by asking an assistant to load up a spreadsheet with corporate data. The CFO would then "play with" the data, trying out various ideas and possibilities and "what if" scenarios. The cycle time between trials would be reduced from days or hours to minutes. The CFO's full, rich information would be applied immediately to the effects of each trial. Unexpected patterns—suggestive to the CFO but often meaningless to a less knowledgeable assistant—would be immediately identified and explored further.

It is generally acknowledged that spreadsheet software that enables expert users to "do it themselves" has led to better outcomes that are achieved faster (Levy 1984; Schrage 2000). The advantages are similar in the case of

product and service development. Learning by doing via trial and error still occurs, of course, but the cycle time is much faster because the complete cycle of need-related learning is carried out at a single (user) site earlier in the development process.

Repartitioning of Development Tasks

To create the setting for a toolkit, one must partition the tasks of product development to concentrate need-related information in some and solution-related information in others. This can involve fundamental changes to the underlying architecture of a product or service. As illustration, I first discuss the repartioning of the tasks involved in custom semiconductor chip development. Then, I show how the same principles can be applied in the less technical context of custom food design.

Traditionally, fully customized integrated circuits were developed in an iterative process like that illustrated in figure 11.1. The process began with a user specifying the functions that the custom chip was to perform to a manufacturer of integrated circuits. The chip would then be designed by manufacturer employees, and an (expensive) prototype would be produced and sent to the user. Testing by the user would typically reveal faults in the chip and/or in the initial specification, responsive changes would be made, a new prototype would be built. This cycle would continue until the user was satisfied. In this traditional manufacturer-centered development process, manufacturers' development engineers typically incorporated need-related information into the design of both the fundamental elements of a circuit—such as transistors, and the electrical "wiring" that interconnected those elements into a functioning circuit.

The brilliant insight that allowed custom design of integrated circuits to be partitioned into solution-related and need-related subtasks was made by Mead and Conway (1980). They determined that the design of a digital chip's fundamental elements, such as its transistors, could be made standard for all circuits. This subtask required rich access to the manufacturer's sticky solution information regarding how semiconductors are fabricated, but did not require detailed information on users' specific needs. It could therefore be assigned to manufacturer-based chip-design and chip-fabrication engineers. It was also observed that the subtask of interconnecting standard circuit elements into a functioning integrated circuit required only sticky,

need-related information about a chip's function—for example, whether it was to function as a microprocessor for a calculator or as a voice chip for a robotic dog. This subtask was therefore assigned to users along with a toolkit that enabled them to do it properly. In sum, this new type of chip, called a gate array, had a novel architecture created specifically to separate the problem-solving tasks requiring access to a manufacturer's sticky solution information from those requiring access to users' sticky need information.

The same basic principle can be illustrated in a less technical context: food design. In this field, manufacturer-based designers have traditionally undertaken the entire job of developing a novel food, and so they have freely blended need-specific design into any or all of the recipe-design elements wherever convenient. For example, manufacturer-based developers might find it convenient to create a novel cake by both designing a novel flavor and texture for the cake body, and designing a complementary novel flavor and texture into the frosting. However, it is possible to repartition these same tasks so that only a few draw on need-related information, and these can then be more easily transferred to users.

The architecture of the pizza pie illustrates how this can be done. Many aspects of the design of a pizza, such as the dough and the sauce, have been made standard. User choice has been restricted to a single task: the design of toppings. In other words, all need-related information that is unique to a particular user has been linked to the toppings-design task only. Transfer of this single design task to users can still potentially offer creative individuals a very large design space to play in (although pizza shops typically restrict it sharply). Any edible ingredient one can think of, from eye of newt to edible flowers, is a potential topping component. But the fact that need-related information has been concentrated within only a single product-design task makes it much easier to transfer design freedom to the user.

The Functionality of Toolkits

If a manufacturer outsources need-intensive design tasks to users, it must also make sure that users have the information they need to carry out those tasks effectively. This can be done via a toolkit for user innovation. Toolkits are not new as a general concept—every manufacturer equips its own engineers with a set of tools suitable for developing the type of products or services it wishes to produce. Toolkits for users also are not new—many users

have personal collections of tools that they have assembled to help them create new items or modify standard ones. For example, some users have woodworking tools ranging from saws to glue which can be used to create or modify furniture—in very novel or very standard ways. Others may have a kit of software tools needed to create or modify software. What is new, however, is integrated toolkits enabling users to create *and* test designs for custom products or services that can then be produced "as is" by manufacturers.

Present practice dictates that a high-quality toolkit for user innovation will have five important attributes. (1) It will enable users to carry out complete cycles of trial-and-error learning. (2) It will offer users a solution space that encompasses the designs they want to create. (3) It will be user friendly in the sense of being operable with little specialized training. (4) It will contain libraries of commonly used modules that users can incorporate into custom designs. (5) It will ensure that custom products and services designed by users will be producible on a manufacturer's' production equipment without modification by the manufacturer.

Learning through Trial and Error

It is crucial that user toolkits for innovation enable users to go through complete trial-and-error cycles as they create their designs. Recall that trial-and-error problem solving is essential to product development. For example, suppose that a user is designing a new custom telephone answering system for her firm, using a software-based computer-telephony integration (CTI) design toolkit provided by a vendor. Suppose also that the user decides to include a new rule to "route all calls of X nature to Joe" in her design. A properly designed toolkit would allow her to temporarily place the new rule into the telephone system software, so that she could actually try it out (via a real test or a simulation) and see what happened. She might discover that the solution worked perfectly. Or she might find that the new rule caused some unexpected form of trouble—for example, Joe might be flooded with too many calls—in which case it would be "back to the drawing board" for another design and another trial.

In the same way, toolkits for innovation in the semiconductor design field allow users to design a circuit that they think will meet their needs and then test the design by "running" it in the form of a computer simulation. This quickly reveals errors that the user can then quickly and cheaply fix

using toolkit-supplied diagnostic and design tools. For example, a user might discover by testing a simulated circuit design that a switch needed to adjust the circuit had been forgotten and make that discovery simply by trying to make a needed adjustment. The user could then quickly and cheaply design in the needed switch without major cost or delay.

One can appreciate the importance of giving the user the capability for trial-and-error learning by doing in a toolkit by thinking about the consequences of not having it. When users are not supplied with toolkits that enable them to draw on their local, sticky information and engage in trial-and-error learning, they must actually order a product and have it built to learn about design errors—typically a very costly and unsatisfactory way to proceed. For example, automobile manufacturers allow customers to select a range of options for their cars, but they do not offer the customer a way to learn during the design process and before buying. The cost to the customer is unexpected learning that comes too late: "That wide-tire option did look great in the picture. But now that the car has been delivered, I discover that I don't like the effect on handling. Worse, I find that my car is too wide to fit into my garage!"

Similar disasters are often encountered by purchasers of custom computers. Many custom computer manufacturers offer a website that allows users to "design your own computer online." However, these websites do not allow users to engage in trial-and-error design. Instead, they simply allow users to select computer components such as processor chips and disk drives from lists of available options. Once these selections have been made, the design transaction is complete and the computer is built and shipped. The user has no way to test the functional effects of these choices before purchase and first field use—followed by celebration or regret.

In contrast, a sophisticated toolkit for user innovation would allow the user to conduct trial-and-error tests to evaluate the effects of initial choices made and to improve on them. For example, a computer design site could add this capability by enabling users to actually test and evaluate the hardware configuration they specify on their own programs and computing tasks before buying. To do this, the site might, for example, provide access to a remote computer able to simulate the operation of the computer that the user has specified, and provide performance diagnostics and related choices in terms meaningful to the user (e.g., "If you add option x at cost y, the time it takes to complete your task will decrease by z seconds"). The user

could then modify or confirm initial design choices according to trade-off preferences only he or she knows.

Appropriate Solution Spaces

Economical production of custom products and services is achievable only when a custom design falls within the pre-existing capability and degrees of freedom built into a particular manufacturer's production system. My colleagues and I call this the *solution space* offered by that system. A solution space may vary from very large to small, and if the output of a toolkit is tied to a particular production system, then the design freedom that a toolkit can offer a user will be accordingly large or small. For example, the solution space offered by the production process of a manufacturer of custom integrated circuits offers a huge solution space to users—it will produce any combination of logic elements interconnected in any way that a user-designer might desire, with the result that the user can invent anything from a novel type of computer processor to a novel silicon organism within that space. However, note that the semiconductor production process also has stringent limits. It will only implement product designs expressed in terms of semiconductor logic—it will not implement designs for bicycles or houses. Also, even within the arena of semiconductors, it will only be able to produce semiconductors that fit within a certain range with respect to size and other properties. Another example of a production system offering a very large solution space to designers—and, potentially to user-designers via toolkits—is the automated machining center. Such a device can basically fashion any shape out of any machinable material that can be created by any combination of basic machining operations such as drilling and milling. As a consequence, toolkits for innovation intended to create designs that can be produced by automated machining centers can offer users access to that very large solution space.

Large solution spaces can typically be made available to user-designers when production systems and associated toolkits allow users to manipulate and combine relatively basic and general-purpose building blocks and operations, as in the examples above. In contrast, small solution spaces typically result when users are only allowed to combine a relatively few pre-designed options. Thus, users who want to design their own custom automobiles are restricted to a relatively small solution space: they can only make choices from lists of options regarding such things as engines, transmissions, and

paint colors. Similarly, purchasers of eyeglasses are restricted to combining "any frame from this list" of pre-designed frames, with "any lens type from that list" of pre-designed options.

The reason producers of custom products or services enforce constraints on the solution space that user-designers may use is that custom products can be produced at reasonable prices only when custom user designs can be implemented by simply making low-cost adjustments to the production process. This condition is met within the solution space on offer. However, responding to requests that fall outside that space will require small or large additional investments by the manufacturer. For example, a producer of integrated circuits may have to invest many millions of dollars and rework an entire production process in order to respond to a customer's request for a larger chip that falls outside the solution space associated with its present production equipment.

User-Friendly Tools

User toolkits for innovation are most effective and successful when they are made "user friendly" by enabling users to use the skills they already have and to work in their own customary and well-practiced design language. This means that users don't have to learn the—typically different—design skills and language customarily used by manufacturer-based designers, and so they will require much less training to use the toolkit effectively.

For example, in the case of custom integrated circuit design, the users of toolkits are typically electrical engineers who are designing electronic systems that will incorporate custom semiconductor chips. The digital design language normally used by electrical engineers is Boolean algebra. Therefore, user-friendly toolkits for custom semiconductor design are provided that allow toolkit users to design in this language. That is, users can create a design, test how it works, and make improvements using only their own, customary design language. At the conclusion of the design process, the toolkit then translates the user's logical design into the design inputs required by the semiconductor manufacturer's production system.

A design toolkit based on a language and skills and tools familiar to the user is only possible to the extent that the user *has* familiarity with some appropriate and reasonably complete language and set of skills and tools. Interestingly, this is the case more frequently than one might initially suppose, at least in terms of the *function* that a user wants a product or service

to perform—because functionality is the face that the product or a service presents to the user. (Indeed, an expert user of a product or service may be much more familiar with that functional face than manufacturer-based experts.) Thus, the user of a custom semiconductor is the expert in what he or she wants that custom chip to *do*, and is skilled at making complex tradeoffs among familiar functional elements to achieve a desired end: "If I increase chip clock speed, I can reduce the size of my cache memory and. . . ."

As a less technical example, consider the matter of designing a custom hairstyle. There is certainly a great deal of information known to hairstylists that even an expert user may not know, such as how to achieve a certain look by means of layer cutting, or how to achieve a certain streaked color pattern by selectively dying some strands of hair. However, an expert user is often very well practiced at the skill of examining the shape of his or her face and hairstyle as reflected in a mirror, and visualizing specific improvements that might be desirable in matters such as curls, shape, or color. In addition, the user will be very familiar with the nature and functioning of everyday tools used to shape hair, such as scissors and combs.

A user-friendly toolkit for hairstyling innovation can be built upon these familiar skills and tools. For example, a user can be invited to sit in front of a computer monitor, and study an image of her face and hairstyle as captured by a video camera. Then, she can select from a palette of colors and color patterns offered on the screen, can superimpose the effect on her existing hairstyle, can examine it, and can repeatedly modify it in a process of trial-and-error learning. Similarly, the user can select and manipulate images of familiar tools, such as combs and scissors, to alter the image of the length and shape of her own hairstyle as projected on the computer screen, can study and further modify the result achieved, and so forth. Note that the user's new design can be as radically new as is desired, because the toolkit gives the user access to the most basic hairstyling variables and tools such as hair color and scissors. When the user is satisfied, the completed design can be translated into technical hairstyling instructions in the language of a hairstyling specialist—the intended production system in this instance.

In general, steady improvements in computer hardware and software are enabling toolkit designers to provide information to users in increasingly friendly ways. In earlier days, information was often provided to users in the form of specification sheets or books. The user was then required to

know when a particular bit of information was relevant to a development project, find the book, and look it up. Today, a large range of potentially needed information can be embedded in a computerized toolkit, which is programmed to offer the user items of information only if and as a development being worked on makes them relevant.

Module Libraries

Custom designs seldom are novel in all their parts. Therefore, a library of standard modules will be a valuable part of a toolkit for user innovation. Provision of such standard modules enables users to focus their creative work on those aspects of their product or service designs that cannot be implemented via pre-designed options. For example, architects will find it very useful to have access to a library of standard components, such as a range of standard structural support columns with pre-analyzed structural characteristics, that they can incorporate into their novel building designs. Similarly, users who want to design custom hairstyles will often find it helpful to begin by selecting a hairstyle from a toolkit library. The goal is to select a style that has some elements of the desired look. Users can then proceed to develop their own desired style by adding to and subtracting from that starting point.

Translating Users' Designs for Production

The "language" of a toolkit for user innovation must be convertible without error into the language of the intended production system at the conclusion of the user's design work. If it is not, the entire purpose of the toolkit will be lost—because a manufacturer receiving a user design will essentially have to do the design work over again. Error-free translation need not emerge as a major problem—for example, it was never a major problem during the development of toolkits for integrated circuit design, because both chip designers and chip producers already used a language based on digital logic. In contrast, in some fields, translating from the design language preferred by users to the language required by intended production systems can be *the* central problem in toolkit design. As an illustration, consider a recent toolkit test project managed by Ernie Gum, the Director of Food Product Development for the USA FoodServices Division of Nestlé.

One major business of Nestlé FoodServices is producing custom food products, such as custom Mexican sauces, for major restaurant chains.

Custom foods of this type have traditionally been developed by or modified by the chains' executive chefs, using what are in effect design and production toolkits taught by culinary schools: recipe development procedures based on food ingredients available to individuals and restaurants, and processed with restaurant-style equipment. After using their traditional toolkits to develop or modify a recipe for a new menu item, executive chefs call in Nestlé Foodservices or another custom food producer and ask that firm to manufacture the product they have designed—and this is where the language translation problem rears its head.

There is no error-free way to translate a recipe expressed in the language of a traditional restaurant-style culinary toolkit into the language required by a food-manufacturing facility. Food factories must use ingredients that can be obtained in quantity at consistent quality. These are not the same as, and may not taste quite the same as, the ingredients used by the executive chef during recipe development. Also, food factories use volume production equipment, such as huge-steam-heated retorts. Such equipment is very different from restaurant-style stoves and pots and pans, and it often cannot reproduce the cooking conditions created by the executive chef on a stovetop—for example, very rapid heating. Therefore, food-production factories cannot simply produce a recipe developed by or modified by an executive chef "as is" under factory conditions—it will not taste the same.

As a consequence, even though an executive chef creates a prototype product using a traditional chef's toolkit, food manufacturers find most of that information—the information about ingredients and processing conditions—useless because it cannot be straightforwardly translated into factory-relevant terms. The only information that can be salvaged is the information about taste and texture contained in the prototype. And so, production chefs carefully examine and taste the customer's custom food prototype, then try to make something that tastes the same using factory ingredients and methods. But an executive chef's taste buds are not necessarily the same as production chef taste buds, and so the initial factory version—and the second and the third—is typically not what the customer wants. So the producer must create variation after variation until the customer is finally satisfied.

To solve the translation problem, Gum created a novel toolkit of preprocessed food ingredients to be used by executive chefs during food development. Each ingredient in the toolkit was the Nestlé factory version of an

ingredient traditionally used by chefs during recipe development: That is, it was an ingredient commercially available to Nestlé that had been processed as an independent ingredient on Nestlé factory equipment. Thus, a toolkit designed for developing Mexican sauces would contain a chili puree ingredient processed on industrial equipment identical to that used to produce food in commercial-size lots. (Each ingredient in such a toolkit also contains traces of materials that will interact during production—for example, traces of tomato are included in the chili puree—so that the taste effects of such interactions will also be apparent to toolkit users.)

Chefs interested in using the Nestlé toolkit to prototype a novel Mexican sauce would receive a set of 20–30 ingredients, each in a separate plastic pouch. They would also be given instructions for the proper use of these ingredients. Toolkit users would then find that each component differs slightly from the fresh components he or she is used to. But such differences are discovered immediately through direct experience. The chef can then adjust ingredients and proportions to move to the desired final taste and texture that is desired. When a recipe based on toolkit components is finished, it can be immediately and precisely reproduced by Nestlé factories—because now the executive chef is using the same language as the factory. In the Nestlé case, field testing by Food Product Development Department researchers showed that adding the error-free translation feature to toolkit-based design by users reduced the time of custom food development from 26 weeks to 3 weeks by eliminating repeated redesign and refinement interactions between Nestlé and purchasers of its custom food products.

Discussion

A toolkit's success in the market is significantly correlated with that toolkit's quality and with industry conditions. Thus, Prügl and Franke (2005) studied the success of 100 toolkits offered in a single industry: computer gaming. They found that success, evaluated by independent experts, was significantly correlated with the quality of execution of the attributes of toolkits that have been discussed in this chapter. That is, success was found to be significantly affected by the quality of trial-and-error learning enabled by a toolkit, by the quality of fit of the solution space offered to users' design problems, by the user friendliness of the tools provided, and by the quality of module libraries offered with the toolkit. Schreier and Franke

(2004) also obtained information on the importance of toolkit quality in a study of the value that users placed on consumer products (scarves, T shirts, cell phone covers) customized with a simple, manufacturer-supplied toolkit. They found user willingness to pay for custom designs, as measured by Vickrey auctions, was significantly negatively affected by the difficulty of creating custom designs with a toolkit. In contrast, willingness to pay was significantly positively affected by enjoyment experienced in using a toolkit.

With respect to industry and market conditions, the toolkit-for-user innovation approach to product design is likely to be most appealing to toolkit suppliers when the heterogeneous needs of *many* users can be addressed by a standard solution approach encoded in a toolkit. This is because it can be costly to encode all the solution and production information relevant to users' design decisions. For example, a toolkit for custom semiconductor design must contain information about the semiconductor production process needed to ensure that product designs created by users are in fact producible. Encoding such information is a one-time cost, so it makes the best economic sense for solution approaches that many will want to use.

Toolkits for user innovation are not an appropriate solution for all product needs, even when heterogeneous needs can be addressed by a common solution approach. Specifically, toolkits will not be the preferred approach when the product being designed requires the highest achievable performance. Toolkits incorporate automated design rules that cannot, at least at present, translate designs into products or software as skillfully as a human designer can. For example, a design for a gate array generated with a toolkit will typically take up more physical space on a silicon chip than would a fully custom-developed design of similar complexity. Even when toolkits are on offer, therefore, manufacturers may continue to design certain products (those with difficult technical demands) while customers take over the design of others (those involving complex or rapidly evolving user needs).

Toolkits can be designed to offer a range of capabilities to users. At the high end, with toolkits such as those used to design custom integrated circuits, users can truly innovate, creating anything implementable in digital electronics, from a dishwasher controller to a novel supercomputer or form of artificial life. At the low end, the product configurators commonly

offered by manufacturers of mass-customized products enable, for example, a watch purchaser to create a custom watch by selecting from lists of predesigned faces, hands, cases, and straps. (Mass-customized production systems can manufacture a range of product variations in single-unit quantities at near mass-production costs (Pine 1993). In the United States, production systems used by these manufacturers are generally based on computerized production equipment.)

The design freedom provided by toolkits for user innovation may not be of interest to all or even to most users in a market characterized by heterogeneous needs. A user must have a great enough need for something different to offset the costs of putting a toolkit to use for that approach to be of interest. Toolkits may therefore be offered only to a subset of users. In the case of software, toolkits may be provided to all users along with a standard, default version of the product or service, because the cost of delivering the extra software is essentially zero. In such a case, the toolkit's capability will simply lie unused in the background unless and until a user has sufficient incentive to evoke and employ it.

Provision of toolkits to customers can be a complement to lead user idea-generation methods for manufacturers. Some users choosing to employ a toolkit to design a product precisely right for their own needs will be lead users, whose present strong need foreshadows a general need in the market. Manufacturers can find it valuable to identify and acquire the generally useful improvements made by lead users of toolkits, and then supply these to the general market. For this reason, manufacturers may find it valuable implement toolkits for innovation even if the portion of the target market that can directly use them is relatively small.

Toolkits can affect existing business models in a field in ways that may or may not be to manufacturers' competitive advantage in the longer run. For example, consider that many manufacturers of products and services profit from both their design capabilities and their production capabilities. A switch to user-based customization via toolkits can affect their ability to do this over the long term. Thus, a manufacturer that is early in introducing a toolkit approach to custom product or service design may initially gain an advantage by tying that toolkit to its particular production facility. However, when toolkits are made available to customer designers, this tie often weakens over time. Customers and independent tool developers can eventually learn to design toolkits applicable to the processes of several

manufacturers. Indeed, this is precisely what has happened in the custom integrated circuit industry. The toolkits revealed to users by the initial innovator, LSI, and later by rival producers were producer-specific. Over time, however, Cadance and other specialist toolkit supply firms emerged and developed toolkits that could be used to make designs producible by a number of vendors. The end result is that manufacturers that previously benefited from selling their product-design skills and their production skills can be eventually forced by the shifting of design tasks to customers via toolkits to a position of benefiting from their production skills only.

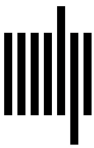
Manufacturers that think long-term disadvantages may accrue from a switch to toolkits for user innovation and design will not necessarily have the luxury of declining to introduce toolkits. If any manufacturer introduces a high-quality toolkit into a field favoring its use, customers will tend to migrate to it, forcing competitors to follow. Therefore, a firm's only real choice in a field where conditions are favorable to the introduction of toolkits may be whether to lead or to follow.

Democratizing Innovation

By: Eric von Hippel

Citation:

Democratizing Innovation
By: Eric von Hippel
DOI: 10.7551/mitpress/2333.001.0001
ISBN (electronic): 9780262285636
Publisher: The MIT Press
Published: 2006



The MIT Press

MIT Press books may be purchased at special quantity discounts for business or sales promotional use. For information, please email special_sales@mitpress.mit.edu or write to Special Sales Department, The MIT Press, 5 Cambridge Center, Cambridge, MA 02142.

Set in Stone sans and Stone serif by The MIT Press. Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Hippel, Eric von.
Democratizing innovation / Eric von Hippel.
p. cm.
Includes bibliographical references and index.
ISBN 0-262-00274-4
1. Technological innovations—Economic aspects. 2. Diffusion of innovations.
3. Democracy. I. Title.
HC79.T4H558 2005
338'.064—dc22 2004061060

10 9 8 7 6 5 4 3 2 1