Complexity is not an inherently negative outcome, but designers need to effectively manage its technical and qualitative facets.

Rob Tannen, Director of Research and Interface Design, Bresslergroup

Designing with Complexity

by Rob Tannen

Philip Glass had his seventy-fifth birthday this year. The work of this polarizing composer, known for his minimalist symphonies and operas, is often perceived (and satirized) for its seeming repetitiveness and redundancy. To some, this translates as elegant subtlety; to others, it is irritatingly melodic simplicity. In fact, although Glass's compositions are often based on repeated phrases, they frequently change and consist of highly varied rhythmic structures with multiple time signatures and polyrhythms. Such music challenges the traditional conventions of musical complexity, in that it is primarily based on rhythm rather than melody. As Glass noted at a recent *New York Times* Arts & Leisure weekend event, "It's the complexity of simplicity that's interesting."

Like music, product design is a creative form in which it is difficult to come to terms on agreed definitions of simplicity and complexity. Even though we encounter and apply these words all the time—in design briefs, product requirements, and advertising (for example, the electronics company Philips and its brand promise of Sense & Simplicity)—we often speak across each other. The terms can be ambiguous because they can apply to the technical attributes of a product, as well as to the characteristics of a user's experience with that product—that is, the quantitative and the qualitative.

To address this ambiguity, we separate the terminology for the objective and the subjective and apply more-accurate and appropriate terms.

Referencing the primary definition of complexity (that is, composed of many interconnected parts), we can refer to the technical aspects of a product in a tangible, objective manner (for example, number of components, time to assemble, and so on). By comparison, the characteristics of user experience are relatively subjective and variable. What is frequently invoked by simplicity—implying that a product is easy to learn, understand, and use—may be more precisely defined as a matter of clarity. Consequentially, a technically simple product can be difficult to understand, while a technically complex product can be easy to understand. In fact, technical complexity is sometimes necessary to achieve clarity. A clever illustration of this concerns the design of web page workflows. Everyone is familiar with the arduous process of filling out online registration forms. Website users find a lengthy, single-page form to be slower to complete when compared to a form that presents information in chunks—an information organization scheme known as progressive disclosure. However, creating the so-called faster form requires greater technical complexity in terms of design, coding, and implementation.

A second ambiguity in describing complexity is whether we are referring to the experience of the product designer or the product user. In other words, on which point in the context of the product's lifecycle are we focusing? There is a common and understandable bias toward focusing on the user's experience. After all, feature fatigue in consumers is directly traceable to the product design process that preceded and defined the products they use. As product designers, we have the opportunity and the responsibility to address this problem, so we must consider the techniques available to us.

Consequentially, a technically simple product can be difficult to understand, while a technically complex product can be easy to understand.

The three *D*'s of designing with complexity

There are several ways to optimize the relationship between the technical complexities of features and the clarity of the user experience. Here I present three general approaches: defeaturing, demystifying, and distributing. These should not be considered exclusive of each other or of other methods. In fact, the thoughtful application and combination of methodologies to come up with best solutions may be the key to effectively designing with complexity.

Defeaturing

The most obvious way to reduce technical complexity and achieve a clearer user experience is to reduce the features within a product. An exemplar of this approach was the line of Flip video cameras. When the Flip camera was first launched in 2007, the consumer video camera market was an escalating feature war. By comparison, the Flip was massively defeatured, providing a total of four controls. Even the built-in USB connector reduced the need for the myriad of cables and connectors found with traditional video cameras.

Of course, the Flip didn't emerge only out of the idea of reducing technical complexity; it also was an answer to changing trends in the use of home video—specifically, the growing trend of uploading brief videos for YouTube and other social network sites. This speaks to the great challenge in defeaturing—achieving agreement from product stakeholders. This challenge is more social science than rocket science—that is, getting buy-in and agreement from a client

and/or project team. User research and voice-of-customer techniques are effective for building a case for defeaturing. It is critical to identify the right features for removal based on understanding customers' needs (rather than needs simply perceived by the design team). For example, in a recent project that I led to redesign a highly complex industrial application, user research revealed that the system provided more functions than were needed by everyday operators, and that a defeatured version would be clearer and more appropriate. Presenting those findings in the voice of the customer (literally, via video recordings) is a particularly powerful way to communicate the message of defeaturing.

Of course, it is not always possible to effectively reduce the number of features in a product, so we turn to demystifying the complexity.

Demystifying

Demystifying is arguably the most important contribution designers bring to the product development process. Fundamentally, it is translating a product's technical complexity into a clear user experience. Demystifying does not reduce the technical complexity of a product; rather, it improves how the user interacts with that complexity.

In recent years, the proliferation of digital interfaces has provided designers with a richly flexible tool set for addressing complexity, even within more-traditional product contexts. Examples of everything from automobiles to ovens to keychains can now be found with embedded interfaces, but the results have not always been positive. In fact, the proliferation of user interfaces clearly demonstrates the difference between technical complexity and experience clarity, in that two products that may be functionally equivalent under the skin can be experienced in different ways, depending on the design of the interface.

Demystifying is arguably the most important contribution designers bring to the product development process.

Case in point—the infamous BMW iDrive fiasco. In the early days of integrating computer interface technologies into the automotive experience (2002, to be precise), the iDrive replaced a multitude of separate physical interface elements with a single knob interface that controlled communications, navigation, and entertainment systems. Unfortunately, a reduced number of control elements paired with an increased number of features led to a usability and public relations failure.[1] In spirit, the multitude of knobs was still there, but it had been replaced with less tangible and intuitive menus that required negotiation and navigation—while driving. BMW subsequently ameliorated the design, primarily by adding more dedicated physical controls.

Superficially, the iDrive appeared easy to use, but the new design had merely displaced and hidden the complexity (and poorly, at that), rather than reduce it. But it does bring up the issue of perceived clarity versus experienced clarity—a distinction realized too late by many BMW customers. In many contemporary designs, we see a gap between expected usability and what we actually experience. For example, many consumers prefer smooth, flat-paneled displays for their visual appeal, but subsequently request the tactile feedback and differentiation of physical controls. In other words, designers are challenged with creating effective designs for criteria that change over time (before and after use)—a difficult situation that may be ultimately solved by control technologies that are more adaptable.

We have seen a parallel in the

1. Ethan Smith, "Driven to Distraction," *New York Times*, Dec. 1, 2002.

popularity of tablets, which are eating away at the traditional laptop computer market. Lacking many physical controls, tablets have greater perceived clarity compared to laptops. This itself is nothing new, as tablets have been around for many years without successful market penetration. It has been the more recent improvement in user interaction that now differentiates the newer generation of tablets from laptops and earlier tablets. Tablets are successful because they effectively combine both defeaturing and demystification. While earlier generations of PC tablets layered tablet features on top of an already complex operating system, new tablets such as the iPad provide limited features via a walled garden. Demystification is achieved by designing the interface specifically for the tablet.

Systems that are not as heavily featured as computers can still benefit from demystification. In 2009, Black & Decker launched its Thermal Leak Detector, with design support from Bresslergroup. The Thermal Leak Detector (Figure 1) is a consumer electronic device that applies infrared technology to measure temperature gradients in order to stop home energy loss. Like the Flip camera, the underlying technology of the leak detector was not new, but the focus on the interface resulted in a clearer user experience. The Thermal Leak Detector uses LED light projection that provides clarity by projecting temperature differences on the targeted spot as color changes (red for hotter, blue for colder), rather than requiring users to read numerical temperature differences on the handheld display. With this product, demystification was achieved not by reducing technical complexity, but by adding a feature that would improve ease of use.

Figure 1. The Black & Decker Thermal Leak Detector improves the clarity of a home task by translating technical data into readily understood color-coded light projections.

(Re)Distributing

Defeaturing reduces technical complexity in the most obvious way—by deleting features. Demystifying reduces it by focusing on how it is presented to the user. The third approach to addressing the relationship between product complexity and the user experience is distribution. Distribution refers to where the complexity resides within the product-user relationship and interaction.

Automation has played an essential role in distributing complexity. A familiar example is the development from manual to automatic transmissions in automobiles. Both types of transmissions provide the same functional result, but in the automatic transmission the technical complexity

of decision-making and the physical process of shifting gears at the appropriate times have been redistributed from the human to an internal mechanism. It's interesting to observe the proliferation of the even more technically complex mixed transmission (for example, the autoshift)—in which the transmission is essentially automated, but the driver can also manually shift when he or she desires. That's a system that fluidly redistributes the complexity based on the driver's preference.

One of the drawbacks of automatic transmissions is that they reduce the driver's connection to the vehicle and the road. This reflects a general challenge in distributing complexity—how to balance the need to provide clear interaction and situational awareness without overloading the user with information about technical complexity. Hence, another critical role for the designer is to determine where to allocate and how to communicate complexity within automation.

This is a challenge my team had to address recently in the design of a new commercial watering system, the UgMO UG1000. Commercial watering systems are technically complex networks of sprinklers, valves, pipes, and controllers for delivering water to maintain golf courses, campuses, and other large, variable green spaces. As we all know, weather systems are extremely complex and difficult to predict. In traditional commercial watering systems, this complexity is handled primarily by the user, who must rely on observation, weather forecasts, and experience to manage a watering schedule for various zones. The UG1000 (Figure 2) redistributes this complexity from the user to the product via a network of wireless moisture sensors embedded into each of the watering zones. The sensors gauge the need for watering and relay it to a central controller, resulting in more-effective watering and healthier grass, eliminating the need to set timers and, most important, conserving water.

Compared to a traditional system, the UG1000 redistributes a high degree of technical complexity from the user to the product, but the result is that the user, who is still ultimately responsible and accountable, risks being significantly removed from the details of operation. We addressed these concerns in the design of the user interface, communicating relevant information, such as moisture levels and trends, so that the user could apply his or her expertise to fine-tune the settings. Actively displaying com-

Figure 2. By embedding wireless moisture sensors into each watering zone, the UgMO UG1000 redistributes the vast complexity of the weather system from the groundskeeper to the product, but effectively keeps the user in the loop.

plexity may seem counterintuitive to designers, but it is frequently critical to do so to keep the user in the loop with complex/automated technologies. The key is to find the balance between awareness and overload. In this case, we had to first distribute the complexity back to the system, then demystify what had been redistributed back to the user in a clear way.

Complexity and product strategy

I've discussed three means for addressing complexity in product design: defeaturing, demystifying, and distributing. These techniques are applicable to optimize the design of any product, and can also be applied to the development and differentiation of a product line or brand strategy.

Many product lines include low, middle, and high-end products within any category. Typically, the low-end products are defined by their reduced feature set and thus are less technically complex. Middle-tier products add additional features; high-end products provide the greatest number of features. This basic additive model may be changing as user experience design gains greater emphasis. For example, middle and high-end products may have equivalent features but differ in the quality of the user experience as delivered by the interface. Here, clarity is a feature unto itself—a markedly distinguishing characteristic in many of Apple's products, for example.

The relationship between feature complexity and usability, and their impact on product success (for instance, revenue), has been analyzed by Debora Thompson and colleagues at the University of Maryland. Although a detailed discussion of this research is beyond the scope of this article, its impact is clear. The authors conducted a series of studies on consumer perception of features.[2] Not surprisingly, increasing the number of product features has a positive effect on perceived capability but a negative effect on perceived usability. Moreover, the impact on usability encourages a negative perception of the company: "As firms' reliance on continuing customer relationships increases, the optimal number of features decreases, implying that products should be made simpler" (p. 442).

Ultimately, what the user needs is a clearer way to access that complexity—or, to paraphrase Philip Glass, it's the simplicity of the complexity that's interesting.

I've alluded to several techniques that design teams can pursue to become better managers of complexity:

- For defeaturing, effectively identifying and prioritizing user needs via user research and communication
- For demystifying, developing skills in usable interface design to provide effective interaction with complexity
- For distributing, understanding the user's mental model so that whatever complexity is removed is available in an accessible and understandable form when needed

As technology progresses, the product development process will become more challenging. A greater number and range of features will be available to add at a lower incremental cost, leading to more-complex products. Complexity itself is not inherently a negative outcome if it is delivered effectively. Ultimately, what the user needs is a clearer way to access that complexity—or, to paraphrase Philip Glass, it's the simplicity of the complexity that's interesting. ■

2. D. Thompson, R. Hamilton, and R. Rust, "Feature Fatigue: When Product Capabilities Become Too Much of a Good Thing." *Journal of Marketing Research*, vol. 42 (November 2005), pp. 431-442.

Reprint #12232TAN50