

Exhibit 1028

Progressive Disclosure

Summary: Progressive disclosure defers advanced or rarely used features to a secondary screen, making applications easier to learn and less error-prone.

By [Jakob Nielsen](#) on December 3, 2006

Topics: [Application Design](#), [Interaction Design](#)

Interaction designers face a dilemma:

- Users want **power**, features, and enough options to handle all of their special needs. (*Everybody* is a special case somehow. For example: Who wants line numbers in a word processor? Millions of users, that's who, including most big law firms.)
- Users want **simplicity**; they don't have time learn a profusion of features in enough depth to select the few that are optimal for their needs.

Progressive disclosure is one of the best ways to satisfy both of these conflicting requirements. It's a simple, yet powerful idea:

1. Initially, show users **only a few** of the most important options.
2. Offer a **larger set** of specialized options upon request. Disclose these secondary features only if a user asks for them, meaning that most users can proceed with their tasks without worrying about this added complexity.

The **print dialog box** is the classic example of progressive disclosure. When you issue the command to print a document, you'll get a dialog box with a small set of choices — mainly, how many copies to print, but possibly a few other variations, such as whether to print the entire document or a subset, and which printer to use. Sadly, print dialog boxes have grown bloated over the past decade, and some applications offer an initial dialog box with highly detailed options that would be better placed in a secondary dialog box.

The initial print dialog box typically contains one or more buttons for advanced options. These buttons lead to secondary dialogs that let users specify rarely used settings, such as scaling and printing the pages in reverse sequence. If the user clicks the advanced options button, the system discloses the additional features.

Applications vs. Websites

Progressive disclosure has long been one of [application design's primary guidelines](#), because most applications have so many commands, features, and options that it makes sense to defer some to a secondary area. Websites have grown so complex that [progressive disclosure is a good idea for many information-rich sites](#) as well. [Deferring secondary material is also a key guideline for mobile design](#).

An ecommerce site, for example, might mention a few key product attributes on the primary [product page](#) and let users click through to a secondary page to see the product's full specifications.

Hypertext provides a simple implementation of progressive disclosure: higher-level pages contain higher-level concepts and simplified descriptions, and lower-level pages fill in the details for those users who want to know everything.

Benefits of Progressive Disclosure

In a system designed with progressive disclosure, the very fact that something appears on the initial display tells users that it's **important**.

For **novice users**, this helps prioritize their attention so that they spend time only on features that are most likely to be useful to them. By hiding the advanced settings, progressive disclosure helps novice users avoid mistakes and saves them the time they would have spent contemplating features that they don't need.

For **advanced users**, the smaller initial display also saves them time because they avoid having to scan past a large list of features they rarely use.

Progressive disclosure thus improves 3 of [usability's 5 components](#): learnability, efficiency of use, and error rate.

You might assume that by initially focusing users' attention on a few core features, they might build a limiting mental model of the system and thus be unable to understand all of their options. [Research says that these are groundless worries](#): people understand a system *better* when you help them prioritize features and spend more time on the most important ones.

Usability Criteria for Progressive Disclosure

Simple as the concept may seem, there are **two things you must get right** when designing for progressive disclosure:

- You must get the right **split between initial and secondary features**. You have to disclose everything that users frequently need up front, so that they have to progress to the secondary display only on rare occasions. Conversely, the primary list can't contain too many options or you'll fail to sufficiently focus users' attention on truly important issues. Finally, the initial display can't contain confusing features or you'll slow down user performance.
- It must be **obvious how users progress** from the primary to the secondary disclosure levels:
 - First, make the **mechanics** of this operation simple. For a website, follow the guidelines for [visualizing links](#). For an application, place the advanced features button in a clearly visible spot.

- Second, label the button or link in a way that sets clear **expectations** for what users will find when they progress to the next level. (In other words, the progression should have strong [information scent](#).)

Regarding the first point, how do you **determine which features the initial disclosure level should contain**? Task analysis and field studies can give you insights into what people need, even if you're working on a new design. If you're improving an existing system, frequency-of-use statistics can help you prioritize the features. For a website, [basic analytics](#) tell you how often people look at various pages. However, you must supplement such analytics with [observational usability testing](#) to discern whether a page gets many hits because users *want* it or because they simply enter the page by *mistake*. For an application, you can get even more detailed usage data by instrumenting the code to record how often people use various features.

In theory, there's no reason why you can't have **multiple levels of progressive disclosure**. Even though the secondary level is for experienced users, there are still limits to how complex it should be; some options might be so specialized or rare that you should relegate them to a tertiary level. In practice, designs that go beyond 2 disclosure levels typically have low usability because users often get lost when moving between the levels.

If you have so many features that you need 3 or more levels, consider [simplifying your design](#). If you can't scale back the complexity, at least chunk your advanced features into groups that make sense, so that users need check only one place and can ignore any areas that they don't need. Use traditional techniques like [card sorting](#) to get the grouping right, and invest extra time in [user testing](#) to ensure that your design supports real task performance, including both common tasks and advanced tasks.

It's also possible to have **multiple secondary displays**, each of which is revealed by a different control on the initial display. The obvious upside to this is that you can accommodate more features without introducing a tertiary level. The obvious

downside is added complexity on the initial level, since users must consider multiple buttons for advanced options.

Because progressive disclosure's goal is to expedite use of the initial display, it's rarely a good idea to offer multiple ways to progress to secondary options. If you must do this, however, at least ensure that you clearly label all choices.

Staged Disclosure: One Step at a Time

Staged disclosure is a variant in which users step through a linear sequence of options, with a subset displayed at each step. **Wizards** are the classic example of staged disclosure. The differences between progressive disclosure and staged disclosure are:

	Progressive Disclosure	Staged Disclosure
Initial display	Core features	Features that users access first in the task sequence
Subsequent display(s)	Secondary features	Features that users access later in the task, even if these features are equally important (or more important)
Do users access subsequent displays?	Usually not — most users get what they need on the initial display	Yes — unless users stop the task before completing the sequence
Navigation between displays	Hierarchical : users start at the initial display and, if necessary, move to the secondary display and then (often) return to the initial display	Linear : users progress through the task one step at a time
Main usability benefit	Learnability : novice users are focused on the most useful features and confusingly advanced features are hidden	Simplicity : each step is simple and its purpose is clear because features that belong to other steps are hidden

Staged disclosure is **useful** when you can divide a task into distinct steps that have little interaction. It is **problematic** when the steps are interdependent and users must alternate between them.

When researching the usability of [46 web-based applications](#), we tested a hotel reservation system that squeezed all the reservation stages onto a single screen. Comparing this design with most hotel sites — which distribute the reservation stages over multiple pages — shows the benefits and drawbacks of staged disclosure:

- The single-screen design worked well when users were trying to decide which hotel room to get because it displayed, on a single screen, the availability and prices for different hotel room categories across various date options considered by the user. Most hotel sites show room categories, prices, dates, and availability across multiple pages, making it more difficult to experiment with various what-if scenarios. People use all of these features together, with frequent back-and-forth modifications, so they should be on 1 screen.
- The single-screen design also caused usability problems because it included a segment for users to enter their address and credit card information. Hotels need such information to complete a booking, but it's not needed during the exploratory phase when users are experimenting with various reservations. Staged disclosure would defer the payment details to a secondary screen. Doing so would leave more space on the first screen for room information and the exploratory interface, thus protecting users from various errors that result when too many options are displayed at the same time.

As this example shows, staged disclosure requires a thorough task analysis to understand which options are used together and which are better thought of as separate stages. The more features you can defer, the simpler your design, but if you divide the task into too many steps, users get bogged down by excess navigation.

The single-screen hotel reservation design would have been better as a 2-screen design — something overlooked by many naïve analyses that consider 1 screen vs. 5 screens to be the only choices. Testing the 1-screen design with real users (which the hotels don't seem to have done) quickly shows that the compromise of a 2-screen solution would be better.

Managing Complexity

Progressive disclosure and staged disclosure are both strategies to manage the [profusion of features](#) and options in modern user interfaces. They are both more than 30 years old and have proven themselves useful in countless applications, and even some websites. Try them — but stay aware of their weaknesses.

Share this article: [Twitter](#) | [LinkedIn](#) | [Email](#)

Copyright © 1998-2020 Nielsen Norman Group, All Rights Reserved.