NGUYÊN LÝ THIẾT KẾ GIAO DIỆN

# Bộ nguyên lý DOET

Đây là bộ nguyên lý được tạo ra bởi nhà khoa học Donald A. Norman trong lúc ông nghiên cứu về vấn đề tương tác giữa con người với máy tính trong dự án gọi là “Design of Everyday Thigns” – DOET (tạm dịch: thiết kế cho những thứ được dùng hằng ngày). Nó được sử dụng để các nhà thiết kế tích hợp và đánh giá xem giao diện của mình có thực sự hữu ích và giúp cho việc tương tác giữa người với máy tính dễ dàng hơn hay không. Nó bao gồm những nhân tố như sau:

## 1. Hoàn tất những yêu cầu từ phía người dùng

Khi thiết kế một phần mềm nào đó, phải tìm hiểu xem người dùng cần sử dụng gì. Ví dụ khi bạn chạy ứng dụng máy tính trong Windows, các trường hợp sử dụng có thể diễn ra sẽ là: cộng 2 số, trừ 2 số, nhân 2 hoặc nhiều số, rút căn, tính lũy thừa. Để được xem là hiệu quả thì giao diện của ứng dụng máy tính này phải đáp ứng được tất cả những tình huống nói trên. Đây cũng là nhân tố quan trọng hàng đầu trong nguyên lý DOET.

## 2. Sử dụng cả “Knowledge in the Head” và “Knowledge in the World”

“Knowledge in the Head” là những kiến thức mà người dùng đã biết, đã có sẵn khái niệm trong đầu của mình mà không cần người thiết kế phải nói ra hoặc đề cập đến trong giao diện phần mềm. Ví dụ, khi thấy một chiếc máy tính, bạn sẽ biết phải gõ bàn phím ra sao, phải sử dụng chuột như thế nào.

“Knowledge in the World” là những kiến thức mà người dùng chưa biết đến, và sẽ cần phải được ghi chú rõ trên giao diện để người dùng học cách sử dụng. Ví dụ, khi bạn lần đầu chạy game, người thiết kế/lập trình viên sẽ trình diễn cho bạn xem cách sử dụng ứng dụng, cách điều khiển nhận vật, bắn thì nhấn nút nào, chạy thì điều khiển phím nào, xoay góc nhìn thì phải làm gì. Tương tự, ứng dụng chuột của Apple tích hợp hướng dẫn sử dụng touchpad đa điểm cho OS X bởi nó không giống với những gì người ta từng biết về cách sử dụng chuột.

## 3. Đơn giản hóa tác vụ của người dùng

Ai cũng thích sự đơn giản, ngay cả khi sử dụng máy tính, smartphone hay tablet. Nếu nhìn vào một ứng dụng mà có hàng đống nút thì chúng ta có thể nghĩ ngay đến chuyện từ bỏ và tìm kiếm ứng dụng khác. Dù cho ứng dụng của bạn viết ra có hay, có chạy nhanh tới cỡ nào đi nữa thì nếu các bước để thực thi một tác vụ nào đó quá phức tạp, người dùng sẽ không bao giờ sử dụng. Chính vì thế lập trình viên/nhà thiết kế cần phải đơn giản hóa chúng. Theo Norman thì có một số cách đơn giản hóa như sau:

### 3.1. Hỗ trợ về mặt tâm lý:

Giúp người dùng cảm thấy bớt “hoảng sợ” khi nhìn vào giao diện. Một ví dụ dễ thấy đó là combobox – tức một nút mà bạn nhấn vào nó sẽ xuất hiện một số tùy chọn trong đó cho bạn. Lập trình viên có thể sử dụng combobox để cho bạn chọn mã vùng nếu cần nhập số điện thoại chẳng hạn.

### 3.2. Giảm tải những gì mà người dùng phải nhớ:

Ví dụ: trong môi trường DOS, để copy thì bạn phải nhớ lệnh cp, nhớ cú pháp của nó, còn ở Windows, chúng ta chỉ việc nhấn phím phải chuột, chọn Copy là xong. Ví dụ khác, khi bạn cần nhập liệu về thời gian, một ô nhỏ với ngày tháng năm, nói chung là cả bộ lịch, sẽ xuất hiện cho bạn chọn ngày. Bạn không cần phải nhớ cụ thể ngày đó là thứ mấy, tuần nào, tháng nào nữa.

### 3.3. Tự động hóa:

Đây là thứ mà chúng ta rất hay gặp. Chẳng hạn như khi bạn đăng kí tài khoản mới trên Yahoo hay Google, hai trang web này sẽ tự xác định vị trí, quốc gia của bạn dựa vào địa chỉ IP. Hoặc như các máy Android, lúc bạn mới mua máy mà có bỏ SIM hoặc kết nối Wi-Fi, thiết bị sẽ tự nhận diện múi giờ và chỉnh sẵn, bạn không cần phải làm nữa. Hoặc nếu bạn nhập mã vùng là 08, chữ “Thành phố Hồ Chí Minh” sẽ tự xuất hiện.

### 3.4. Thay đổi bản chất của tác vụ:

Trên các trình duyệt cũ, ô tìm kiếm sẽ nằm tách biệt với ô địa chỉ. Sau này, Chrome, Firefox, Safari, IE đều sử dụng chung một ô cho cả hai tác vụ, dễ dàng hơn và bạn sẽ không cần phải suy nghĩ nên nhập liệu vào đâu.

## 4. Làm mọi thứ rõ ràng nhất có thể

Với những tính năng chính, hãy đem nó ra giao diện chính, nơi người dùng dễ thấy nhất. Đừng cố giấu nó vào những menu ẩn sâu trong ứng dụng, điều đó không giúp gì cho người dùng cả. Nó còn làm người dùng rối và cảm thấy bực mình nữa. Việc làm mọi thứ trở nên rõ ràng cũng là một cách nhắc cho người dùng biết những tính năng nào hệ thống của chúng ta có hỗ trợ, và cách dùng chúng như thế nào.

## 5. Thu hẹp khoảng cách thực thi và khoảng cách đánh giá

Lúc mới tiếp cận với một hệ thống, phần mềm nào đó, người dùng sẽ đối mặt với hai “khoảng cách” (Gulf) khiến họ chưa thể dùng hệ thống một cách thành công, đó là:

### 5.1 Khoảng cách thực thi (Gulf of Execution):

Người dùng cần làm gì để một tính năng có thể chạy được? Để giảm khoảng cách giữa sự thắc mắc của người dùng với giao diện, nhà thiết kế có thể dùng một thứ gọi là “Affordance”. Ví dụ, khi bạn thấy một cái nút, bạn sẽ biết là bạn cần phải nhấn vào đó. Hoặc với một thanh trượt (slider), bạn sẽ biết là bạn cần kéo con chạy để thay đổi giá trị, và kéo sang phải là tăng giá trị lên, còn kéo sang trái là giảm đi.

### 5.2 Khoảng cách đánh giá (Gulf of Evaluation):

Làm sao người dùng biết rằng tính năng đó đã chạy theo đúng những gì nó được thiết kế? Trong trường hợp, chúng ta có thể lấy sự phản hồi xúc giác trên các smartphone làm ví dụ. Khi bạn gõ bàn phím hoặc nhấn nút nào đó, smartphone sẽ rung nhẹ lên để báo hiệu rằng bạn đã thành công trong chạm vào nút đó. Âm thanh phát ra cũng là một sự phản hồi được sử dụng rộng rãi. Nhiệm vụ của nhà thiết kế/lập trình viên sẽ là giúp người dùng trả lời hai câu hỏi nói trên một cách rõ ràng, dễ dàng nhất có thể.

## 6. Sắp xếp một cách đúng đắn

Trong đầu người dùng sẽ có sẵn một mô hình về cách hoạt động của một thứ gì đó. Ví dụ, khi bạn mua đồ trong siêu thị, các bước bạn sẽ làm đó là xem hàng > xem giá > chọn hàng > tính tiền. Áp dụng nó vào giao diện của một trang web bán hàng online, bạn không thể bắt người dùng xem giá rồi mới xem hàng được. Bạn phải theo đúng trình tự và mô hình mà người dùng suy nghĩ đến, khi đó thì việc sử dụng giao diện mới đạt hiệu quả cao.

## 7. Khai thác sức mạnh từ các biện pháp hạn chế

Biện pháp hạn chế (constrain), là một sự tinh chỉnh nào đó vào giao diện để một thứ gì đó không còn sử dụng được nữa, tạm thời hoặc trong thời gian dài. Vì sau chúng ta lại muốn đưa hạn chế vào trong ứng dụng, sao không để người dùng muốn làm gì thì làm? Đó là vì chúng ta muốn người dùng chọn được một lựa chọn chính xác nhất, giảm lỗi về sau.

Có 3 loại biện pháp hạn chế thường sử dụng ngày nay:

***Hạn chế vật lý***: Ví dụ, nhiều phần mềm hiện sử dụng các ô chọn radiobutton để lựa chọn một thứ gì đó, như giới tính chẳng hạn. Bạn chỉ có thể chọn một trong hai, hoặc “Nam” hoặc “Nữ” chứ bạn không thể gõ bàn phím để nhập được. Như vậy sẽ tránh tình huống gõ nhầm, gõ không đúng ý muốn của lập trình viên.

***Hạn chế về suy nghĩ***: ở một giao diện cho phép nhập thông tin cá nhân, nếu ô đầu tiên có nhãn là “Họ”, ô thứ hai ghi chữ “Tên”, bạn sẽ biết cần gõ gì vào. Còn nếu hai ô đó không có nhãn thì bạn có thể nhập tên vào ô Họ, họ và ô Tên, sai quy tắc yêu cầu của phần mềm.

***Hạn chế về văn hóa***: cái này hơi phức tạp, nhưng ví dụ đơn giản nhất là màu sắc. Nếu bạn thấy một nút tô màu đỏ, bạn sẽ nghĩ ngay đến chuyện nhấn vào nút đó có thể gây ra nguy hiểm gì đó. Còn nếu thấy nút màu xanh, bạn biết nút đó là để chấp thuận và tiếp tục. Cách chọn màu cũng phải tùy vào văn hóa, chẳng hạn như ở Nga, người ta xem màu đỏ là “tích cực”, khi đó bạn không nên tô nút “Xóa” với màu này.

## 8. Thiết kế phải tính đến lỗi

Cứ con người là phải mắc lỗi. Đó là chuyện quá hiển nhiên, và phần mềm phải được thiết kế để đảm đương lỗi từ phía người dùng, và cả lỗi phát sinh do bản thân ứng dụng nữa. Thực chất điều cốt lõi ở đây đó là phải loại bỏ lỗi từ gốc và không cho nó xảy ra. Nhưng không thứ gì là hoàn hảo, do đó, khi gặp lỗi thì lập trình viên sẽ phải cung cấp cho người dùng một thông báo hợp lý, ít nhất cũng phải chỉ dẫn cho người dùng nên làm gì để khắc phục lỗi, hoặc thông báo rằng lỗi đó là gì. Nếu được, nên kèm theo là một cách phục hồi hiệu quả để người dùng thoát khỏi lỗi đó.

*Ví dụ 1:* bạn vừa soạn xong bài báo cáo 50 trang trong Word, bỗng nhiên máy tính bị điên, Word bị đóng, thế là bạn mất hết bài viết dài vì chưa kịp lưu. May mắn là Microsoft có tích hợp tính năng AutoSave và khả năng tự khôi phục nên bạn có thể cứu được nội dung của mình.

*Ví dụ 2:* người dùng nhập liệu sai, ô số điện thoại thì chỉ được nhập số nhưng họ nhập vào cả chữ cái nữa. Lúc đó thông báo hiện ra nên là “Bạn chỉ được nhập số, hãy thực hiện lại nhé!”. Nếu chỉ có thông báo “Sai rồi!” thì người dùng chẳng thể biết họ sai cái gì và nên làm thế nào cho đúng.

*Ví dụ 3:* khi bạn đang download một tập tin 50GB, khi tải được 99% thì bạn lỡ nhấn nút đóng cửa sổ trình duyệt. Nếu đóng luôn, bạn sẽ mất hết dữ liệu đã tải. Nhưng may mắn là các hãng trình duyệt đều tích hợp một hộp thoại thông báo rằng “bạn đang download file, đóng trình duyệt là sẽ mất đi đó. Có đồng ý hay không”. Lúc này, nếu bạn chỉ nhấn nhầm vào nút đóng thì có nhấn nút Hủy để quay lại.

## 9. Chuẩn hóa khi cần thiết

Có những thứ mà người ta thường làm rất lộn xộn và không theo quy tác nào cả. Nếu cứ mỗi một phần mềm, một giao diện ứng dụng một kiểu thì người dùng sẽ rất rối. Đây là lúc chúng ta cần một chuẩn chung để tạo ra sự thống nhất cao đó. Chẳng hạn như với Windows, ở thanh menu của bất kì một ứng dụng nào đó thì nút đầu tiên sẽ là File, cuối cùng là Help. Trên máy Mac của Ứng dụngle cũng y như thế. Hay như trên Android, Google đưa ra một chủ đề gọi là “Holo” với hai màu đen và xanh dương đặc trưng để lập trình viên áp dụng thiết kế này cho ứng dụng của mình.

# Bộ nguyên lý Nielsen

Jakob Nielsen năm 1995 ra mắt một tập các nguyên lý mà ông đã nghiên cứu và chứng minh rằng sẽ tốt cho một giao diện khi thiết kế theo các quy chuẩn này.

Về cơ bản thì 10 điều do Nielsen đưa ra cũng khá giống với bộ DOET nói trên, chỉ khác một chút về cách diễn đạt và cách người ta đánh giá nó.

## 1. Luôn hiện rõ trạng thái của hệ thống

Giao diện phải thông báo cho người dùng biết hệ thống đang làm gì, trạng thái như thế nào, công việc đang diễn tiến đến đâu, và thông báo đó phải xuất hiện trong một thời gian hợp lý. Một ví dụ mà chúng ta rất hay thấy đó là thanh tiến trình (progress bar) xuất hiện khi copy file dù cho bạn đang OS X hay Windows, Linux. Lúc bạn mở trang web có thanh màu chạy để biểu thị % trang web đã tải xong cũng là một ví dụ khác. Thời gian hợp lý ở đây có nghĩa là sau khi bạn copy và ra lệnh paste thì thanh tiến trình sẽ hiện ra ngay, chứ không phải 5 hay 10 phút sau mới hiện diện.

## 2. Làm giao diện giống với hiện thực đời sống

Giao diện cần nói phải “nói cùng loại ngôn ngữ” với người dùng, từ cách dùng từ, đặt câu, cách thông báo, nhãn, ghi chú đều phải dễ hiểu với đối tượng người dùng mà giao diện nhắm đến.

Tác vụ phải được thực hiện theo đúng trình tự mà người dùng kì vọng trong đời sống. Ví dụ, nếu bạn đăng kí vé máy bay thì ở ngoài đời bạn sẽ cần phải đến phòng vé, nói tên, nói nước muốn bay đến, nói ngày tháng năm và giờ giấc. Khi lên website đặt hàng thì trình tự này cũng cần phải được giữ lại (có thể tinh chỉnh một hai điểm) để giúp việc sử dụng được dễ dàng và trực quan hơn.

## 3. Cho người dùng sự kiểm soát và sự tự do

Người dùng thường hay nhầm lẫn, và người thiết kế giao diện phải cung cấp một “lối thoát” trong trường hợp người dùng lỡ làm sai một thứ gì đó. Ví dụ dễ thấy là nút Undo và Redo trong hàng loạt ứng dụng, từ trình duyệt, Word, Excel cho đến Notepad và các app trên di động.

## 4. Đề cao sự thống nhất

Trong một hệ thống, những thứ có tính năng tương tự nhau cần phải thống nhất với nhau về màu sắc, vị trí, cách hiển thị, chữ viết,… Chẳng hạn như trên Windows, toàn bộ nút đóng cửa sổ đều màu đỏ với dấu chéo trắng và nằm ở góc trên bên phải của các cửa sổ. Hoặc trong Microsoft Office, toàn bộ những công cụ định dạng văn bản như Bold, Italic, chỉnh font chữ, cỡ chữ đầu nằm ở thẻ Home, bất kể bạn đang dùng Word, Excel, PowerPoint, Visio hay gì đi nữa. Bằng cách thống nhất, chúng ta giúp người dùng tương tác với giao diện một cách nhanh chóng hơn, đỡ phải nhớ nhiều tình huống khác nhau. Với HTC, tất cả app của họ đều có nút tìm kiếm, add và menu ở cạnh trên

## 5. Hạn chế lỗi

Cũng như DOET, hệ thống và giao diện cần phải được thiết kế với tâm niệm rằng người dùng luôn mắc lỗi, và điều quan trọng giao diện là phải bao quát được hầu hết các lỗi thông dụng. Người thiết kế/lập trình viên cũng cần phải hạn chế tối đa nguồn phát sinh lỗi.

Ví dụ: Combobox, radio button, checkbox: chỉ phép người dùng chọn một số thứ do lập trình viên hoặc nhà thiết kế định trước, không thể tự mình nhập cái mới (ví có khả năng phát sinh lỗi) Vô hiệu hóa một thành phần giao diện nào đó khi không cần thiết.

## 6. Cung cấp sự linh hoạt cho người dùng

Lúc mới sử dụng máy tính, để copy một thứ gì bạn thường làm gì? Chọn nội dung hoặc file đó, nhấp phím phải chuột, chọn copy, qua chỗ khác, lại nhấn phím phải chuột, chọn paste. Về sau, khi chúng ta có kinh nghiệm hơn, thao tác đó được đơn giản hóa đi rất nhiều bằng cách sử dụng phím tắt Control + C, Control + V. Đây chính là ví dụ cho nguyên lí thứ 6 của Nielsen.

Ngoài ra, khi hệ thống cần phải thay đổi giao diện, việc giữ lại sự linh hoạt như bản cũ là điều cần thiết, nếu không người dùng sẽ cảm thấy rất thất vọng vì mất đi một thứ mà họ đang dùng quen tay.

## 7. Giảm tải những gì người dùng cần phải nhớ

Để làm được điều này, nhà thiết kế và lập trình viên cần đảm bảo rằng người dùng có thể thấy hết các tính năng chính một cách dễ dàng. Người dùng không cần phải nhớ những thứ phức tạp khi sử dụng các thành phần trong hệ thống. Lập trình viên và nhà thiết kế cần lưu ý rằng việc nhìn và nhận diện (recognition) sẽ luôn tốt hơn là bắt người dùng nhớ lại những gì đã từng diễn ra (recall). Việc ứng dụng biểu tượng đồ họa, gợi ý về password, sử dụng màu sắc để thông báo một thứ gì đó đã được hoàn tất… là những ví dụ cho nguyên lý thứ 7 bởi nó giúp người dùng biết ngay rằng tính năng đó là gì và để làm gì.

Cũng như khi giao diện đồ họa ra mắt, nó đã giúp mở ra cả một kỉ nguyên máy tính mới và ai cũng có thể xài PC, chứ lúc còn chạy bằng giao diện dòng lệnh (CLI) thì không mấy người có thể tiếp cận với máy tính vì nó quá khó và phải nhớ quá nhiều thứ. GUI\_vs\_CLI. Bạn thích dùng cái nào hơn? Giao diện người dùng đẹp mắt hay là giao diện dòng lệnh đen thui và phải gõ gõ tùm lum? Một ví dụ khác: để tìm một sinh viên nào đó thì nên dùng tên chứ đừng bắt buộc người dùng phải nhớ mã sinh viên, mặc dù chúng ta có thể cung cấp thêm tùy chọn tìm kiếm bằng mã.

## 8. Giao diện phải có tính thẩm mĩ, đẹp và có liên quan

Đẹp, đó là chuyện mà ai cũng muốn, và giao diện đẹp thì luôn được đánh giá cao. Hãy nhìn vào cách mà Apple thiết kế iOS hay những gì Google làm với Android, tất cả đều xoay quanh một chữ đẹp. Khi sử dụng một thứ gì đó có tính thẩm mĩ cao thì người dùng sẽ cảm thấy hài lòng hơn và có hứng hơn. Ngoài ra, thông tin cung cấp cho người dùng cũng phải liên quan, nếu không thì cũng bằng thừa.

Ví dụ, trong một thông báo hiện ra trước khi lưu tập tin, thứ mà chúng ta nên đưa vào đó là câu hỏi “bạn có muốn lưu đè tập tin hay không” hoặc “bạn có muốn lưu nó vào đây hay không”, không phải là một bài viết dài về cách lưu tập tin, tác hại khi không xài phần mềm đúng cách (!?).

## 9. Giúp người dùng xác định và cách khắc phục sau khi lỗi xảy ra

Thông báo khi có lỗi phải dễ hiểu và cho người dùng biết lỗi đó là gì, không nên hiện một đống mã lỗi lên. Điều đó sẽ làm người dùng nản chí. Ngoài ra, giao diện cũng cần nói cho người dùng biết những biện pháp khả thi để khắc phục lỗi, chứ không nên để mặc người dùng muốn làm gì thì làm. Bsod\_nt. Thông báo lỗi “màn hình xanh chết chóc” của Windows 7 quá rồi, làm người ta hoảng và không biết phải làm gì hoặc chuyện gì sẽ xảy ra. Còn lên Windows 8, Microsoft thay đổi nó để nói cho chúng ta rằng “hệ điều hành đang thu thập lỗi và sẽ tự khởi động máy lại cho bạn”.

## 10. Cung cấp sự trợ giúp, tài liệu hướng dẫn sử dụng

Bạn có thể thấy rằng bất kỳ phần mềm nào, dù đơn giản đến đâu thì cũng đều có mục trợ giúp (Help) hoặc tài liệu hướng dẫn sử dụng (documentation), hoặc ít nhất cũng có một dòng nói rằng cần sử dụng phần mềm như thế nào. Đây là cách cuối cùng mà người dùng sẽ tìm đến khi họ không thể thực hiện một thao tác nào đó. Hãy thử tưởng tượng xem nếu như bạn sử dụng phần mềm mà khi gặp lỗi, bạn không thể tìm được bất kỳ sự trợ giúp nào thì điều đó thật là tệ. Tùy theo độ phức tạp của giao diện và phần mềm mà người ta sẽ soạn thảo các tài liệu hướng dẫn sử dụng và phần trợ giúp tương ứng. Tooltip box – dòng thông báo nhỏ hiện ra khi chúng ta rê chuột lên một nút, một ô, một hộp thoại,… – cũng chính là một sự trợ giúp chứ không phải cứ vào Help mới là trợ giúp.