**CÂU HỎI ÔN THI MÔN HỌC**

**THIẾT KẾ GIAO DIỆN NGƯỜI SỬ DỤNG**

[1. Nêu và phân tích sự khác nhau giữa một hệ số (factor) và yêu cầu trong thiết kế giao diện người sử dụng? (Chương 1 - Slide 11) 1](#_Toc470787052)

[2. Nêu và phân tích sự khác nhau giữa kiểm thử tính khả dụng, đánh giá heuristic và người dùng đánh giá? (Chương 1) 2](#_Toc470787053)

[3. Nêu và phân tích sự khác nhau giữa mô hình xử lý lặp và mô hình xử lý thác nước (waterfall) trong thiết kế giao diện người sử dụng? (Chương 2 - Slide 7) 4](#_Toc470787054)

[4. Mô tả và phân tích luật hình ảnh (gestalt)? (Chương 3 – Slide 7) 6](#_Toc470787055)

[5. Mô tả và phân tích mô hình ánh xạ? (Chương 4 – Slide 20) 6](#_Toc470787056)

[6. Nêu và phân tích 6 nguyên lý cơ bản trong thiết kế giao diện người sử dụng? (Chương 4) 6](#_Toc470787057)

[7. Nêu và phân tích sự khác nhau giữa mô tả công việc (task) và vùng làm việc (work area)? (Chương 5 - Slide 5) 7](#_Toc470787058)

[8. Nêu và phân tích sự khác nhau giữa công việc (task) và user case? (Chương 5) 9](#_Toc470787059)

[9. Nêu và phân tích sự khác nhau giữa cửa sổ ảo (virtual windows) và cửa sổ (windows) trong màn hình thiết kế (screens)? (Chương 6) 9](#_Toc470787060)

[10. Phương pháp kiểm tra CREDO là gì? Nêu ví dụ minh họa? (Chương 6 - Slide 37) 13](#_Toc470787061)

[11. Tại sao cần hàm điều hướng (navigation) và hàm undo (hoàn tác)? Nêu ví dụ? (chương 7 - Slide 16) 13](#_Toc470787062)

[12. Phân biệt giữa khiếm khuyết phần mềm và yêu cầu thay đổi phần mềm, ví dụ và phương án xử lý? (Chương 8 – Slide 13, 14, 24, 25, 31) 14](#_Toc470787063)

# Nêu và phân tích sự khác nhau giữa một hệ số (factor) và yêu cầu trong thiết kế giao diện người sử dụng? (Chương 1 - Slide 11)

**\* Yêu cầu trong thiết kế giao diên người dùng:** Thiết kế giao diện phải phụ thuộc vào yêu cầu, kinh nghiệm và khả năng của người sử dụng hệ thống.

Người thiết kế cũng nên quan tâm đến những giới hạn vật lý và tinh thần của con người và nên nhận ra rằng con người luôn có thể gây ra lỗi. Không phải tất cả các nguyên tắc thiết kế giao diện đều có thể được áp dụng cho tất cả các giao diện. Cụ thể khi thiết kế sẽ phải áp dụng một số nguyên tắc sau:

- Sự quen thuộc của người sử dụng: giao diện phải được xây dựng dựa trên các thuật ngữ và các khái niệm mà người sử dụng có thể hiểu được hơn là những khái niệm liên quan đến máy tính. Ví dụ: hệ thống văn phòng nên sử dụng các khái niệm như thư, tài liệu, cặp giấy … mà không nên sử dụng những khái niệm như thư mục, danh mục …

- Thống nhất: hệ thống nên hiển thị ở mức thống nhất thích hợp. Ví dụ: các câu lệnh và menu nên có cùng định dạng …

- Tối thiểu hoá sự bất ngờ: nếu một yêu cầu được xử lý theo cách đã biết trước thì người sử dụng có thể dự đoán các thao tác của những yêu cầu tương tư.

- Khả năng phục hồi: hệ thống nên cung cấp một số khả năng phục hồi từ lỗi của người sử dụng và cho phép người sử dụng khôi phục lại từ chỗ bị lỗi. Khả năng này bao gồm cho phép làm lại, hỏi lại những hành động như xoá, huỷ …

- Hướng dẫn người sử dụng: như hệ thống trợ giúp, hướng dẫn trực tuyến …

- Tính đa dạng: hỗ trợ nhiều loại tương tác cho nhiều loại người sử dung khác nhau. Ví dụ: nên hiển thị phông chữ lớn với những người cận thị.

Tương tác giữa người sử dụng và hệ thống được chia thành 5 loại sau:

- Vận hành trực tiếp

- Lựa chọn menu

- Điền vào biểu mẫu (Form)

- Ngôn ngữ ra lệnh

- Ngôn ngữ tự nhiên

**\* Sự khác nhau giữa một hệ số (factor):**

a Hệ thống có thể hỗ trợ các công việc mà người dùng có trong thực tế đời sống.

b Dễ học tập. Làm thế nào dễ dàng là hệ thống để tìm hiểu cho các nhóm khác nhau của người sử dụng?

c Nhiệm vụ hiệu quả. Làm thế nào có hiệu quả là nó cho người sử dụng thường xuyên?

d Dễ nhớ. Làm thế nào là nó dễ dàng để nhớ cho người sử dụng thường xuyên?

e Sự hài lòng chủ quan. Làm hài lòng là người sử dụng với hệ thống?

f Dễ hiểu. Làm thế nào là nó dễ dàng để hiểu những gì hệ thống không? Yếu tố này là đặc biệt quan trọng trong các tình huống bất thường, cho các tình huống lỗi dụ hoặc lỗi hệ thống. Chỉ có một sự hiểu biết về những gì hệ thống không thể giúp người sử dụng ra.

=> Dễ sử dụng (hoặc người sử dụng thân thiện) là sự kết hợp của các yếu tố b đến f.

# Nêu và phân tích sự khác nhau giữa kiểm thử tính khả dụng, đánh giá heuristic và người dùng đánh giá? (Chương 1)

**\* Kiểm thử tính khả dụng (Usability testing)**

Trong một thử nghiệm khả năng sử dụng, chúng ta để cho một người sử dụng (các đối tượng thử nghiệm hoặc sử dụng thử nghiệm) cố gắng để thực hiện các nhiệm vụ thực tế bằng cách sử dụng hệ thống thực hoặc một mô hình của nó.

Có nhiều hướng khác nhau để làm điều đó: Một kỹ thuật được ưa chuộng là yêu cầu người dùng phải suy nghĩ lớn - giải thích những gì anh đang làm gì và tại sao.

**Sử dụng hệ thống Real.**

Bạn có thể muốn tìm các vấn đề khả năng sử dụng trong một hệ thống được hoàn thành hoặc ít nhất là làm việc với một mức độ lớn.

**Nhiệm vụ kiểm tra**

Bạn cũng phải chọn một số nhiệm vụ kiểm tra hoặc tình huống mà người dùng sẽ sử dụng hệ thống.

Ví dụ: nếu chúng ta muốn thử nghiệm một hệ thống để tiếp khách của khách sạn, một nhiệm vụ thử nghiệm tốt sẽ đặt phòng cho khách.

Có rất nhiều nhiệm vụ khác mà phải được kiểm tra quá, tất nhiên.

Lựa chọn các nhiệm vụ kiểm tra bên phải là một vấn đề quan trọng nếu chúng ta muốn tìm tất cả các vấn đề khả năng sử dụng.

**Một nhiệm vụ kiểm tra tốt phải đáp ứng các tiêu chí:**

Một cái gì đó người sử dụng sẽ làm gì trong tình huống công việc thực tế.

Một mảnh đầy đủ các công việc có ý nghĩa (một nhiệm vụ đóng). Đặt trước là tốt - nhân viên tiếp tân xử lý cuộc gọi của khách hàng. Đăng nhập không phải là một nhiệm vụ tốt. Nhân viên lễ tân đã không thực hiện bất cứ điều gì bằng cách đăng nhập.

Nói mà không cần sự giúp đỡ ẩn - mà không cần gợi ý về cách để thực hiện nhiệm vụ.

**Thực hiện phép thử:**Giải thích mục đích:

Tìm vấn đề khi sử dụng hệ thống

Lỗi hệ thống

Hãy cho nhiệm vụ - nghĩ lớn, xin vui lòng

Quan sát, lắng nghe, ghi lại

Hãy hỏi một cách thận trọng:

Bạn đang tìm kiếm cái gì?

Tại sao …?

Giúp người dùng đăng nhập khi họ là chắc chắn bị mất mật khẩu

**\*Báo cáo**

Sau khi kiểm tra, bạn phải viết một danh sách các vấn đề mà người dùng gặp phải. Họ phải có hình thức như vậy mà mọi người biết hệ thống hiểu những gì bạn viết. Bạn phải ghi danh sách này càng sớm càng tốt, tốt nhất là trong vòng 12 giờ. Nếu không, bạn có thể không được nữa hiểu viết nguệch ngoạc của riêng bạn và nhớ những gì đã xảy ra trong quá trình thử nghiệm.

**Heuristic đánh giá & đánh giá người dùng**

Kiểm thủ tính khả dụng có vẻ cồng kềnh. Không có cách nào khác để xác định các vấn đề về khả năng sử dụng? Có lẽ chúng ta có thể để cho một cái nhìn chuyên gia khả năng sử dụng vào màn hình và chỉ ra những vấn đề này? Cách tiếp cận này được gọi là đánh giá heuristic: cách tiếp cận này thường được gọi là xem xét sử dụng.

Đánh giá heuristic có thể được thực hiện bằng nhiều cách:

- Chúng ta có thể để cho các chuyên gia khả năng sử dụng nhìn vào màn hình, hoặc chúng tôi có thể làm cho nhà phát triển đồng nhìn họ.

- Họ có thể sử dụng bản án âm nhạc của họ và kinh nghiệm trước đó, hoặc họ có thể sử dụng một danh sách hướng dẫn heuristic.

- Hướng dẫn có thể, ví dụ, nói rằng các màn hình không nên quá đông đúc, các thông báo lỗi nên cho người dùng phải làm gì, phải có tính năng cho suy vi, vv

- Mỗi đánh giá có thể cung cấp danh sách của riêng mình của các khuyết tật, hay hoặc chúng tôi có thể yêu cầu tất cả trong số họ để đạt được thỏa thuận và đưa ra một danh sách chung.

Trong nhiều trường hợp đánh giá heuristic, phát hiện rất nhiều vấn đề, nhưng khoảng một nửa trong số họ là sai trong ý nghĩa rằng họ không gây ra vấn đề cho người dùng thực sự. Nó sẽ là một sự lãng phí thời gian cố gắng để sửa chữa những vấn đề sai, nhưng chúng ta không biết được những vấn đề này là sai.

Trong những trường hợp này, đánh giá heuristic, nhớ về một nửa của vấn đề nghiêm trọng mà người sử dụng thật gặp phải.

**Người dùng đánh giá (User review)**

Một đánh giá thường được thực hiện với người sử dụng chuyên gia người biết khu vực ứng dụng rất tốt.

Đi qua các màn hình với những người sử dụng chuyên gia, cho thấy làm thế nào họ sẽ được sử dụng, và thảo luận về cách nhiệm vụ khác nhau được thực hiện, những gì là mất tích, những gì là cồng kềnh.

Đánh giá là quan trọng cho việc tìm kiếm chức năng còn thiếu. Người sử dụng chuyên gia có thể trỏ đến nhiều điều hệ thống không thể làm được.

Tuy nhiên, đánh giá là không tốt cho việc tìm kiếm dễ dàng của sử dụng vấn đề.

Người sử dụng chuyên gia thường biết quá nhiều để gặp tất cả các tính năng dễ sử dụng vấn đề mà có thể chặn người dùng điển hình.

Nó sẽ được tốt hơn để xem xét với người dùng bình thường?

Cách nhất thiết kế tiến hành đánh giá giấu tính dễ sử dụng vấn đề.

Các nhà thiết kế giải thích cách hoạt động và người sử dụng không cần phải tìm ra cho mình.

Chúng ta thường thấy các nhà thiết kế trình bày một hệ thống mới cho một nhóm người dùng và các nhà quản lý.

# Nêu và phân tích sự khác nhau giữa mô hình xử lý lặp và mô hình xử lý thác nước (waterfall) trong thiết kế giao diện người sử dụng? (Chương 2 - Slide 7)

* **Mô hình lặp:**

**Study users**

**and tasks**

**HCI classic:**

**iterative design**

* Là một tiến trình lặp gắn kết người sử dụng
* Gồm 3 hoạt động:

+ Phân tích người dùng: Hiểu người dùng làm gì với hệ thống

+ Xác định bản mẫu để thử nghiệm

+ Đánh giá kết quả kiểm thử tính khả dụng

* Điểm yếu:

+ Không có phương pháp thiết kế màn hình

+ Không có phương pháp để sửa chữa các vấn đề về khả năng sử dụng khi chúng được phát hiện

+ Khó có thể tìm thấy người dùng thử nghiệm với cấu hình chuẩn

* **Mô hình thác nước:**

- Pha phân tích: Phân tích yêu cầu chức năng của người sử dụng, khách hàng 🡪 đưa ra giải pháp

- Pha thiết kế: Lập kế hoạch phát triển, đưa ra phướng án thiết kế

- Pha lập trình: Chia team, nhóm xây dựng phần mềm

- Pha kiểm thử: Kiểm tra chương trình chạy trên máy tính, sửa lỗi.

+ Kiểm thử nội bộ: Đơn vị xây dựng phát triển phần mềm kiểm tra, phát hiện lỗi để sửa chữa

+ Kiểm thử khách hàng: Chức năng có đáp ứng được yêu cầu của khách hàng? Chạy phần mềm

* **Sự khác nhau giữa mô hình lặp và mô hình xử lý thác nước:**

- Với mô hình lặp giảm rủi ro sớm trong chu kỳ phát triển phần mềm. Những yêu cầu quan trọng thường được phát triển và chuyển đến người sử dụng sớm.

- Phản hồi của nguời sử dụng về những vấn đề phát sinh trong phiên bản trước được dùng để cải tiến và ngăn ngừa những vấn đề tương tự xảy ra trong những phiên bản tiếp theo.

- Người sử dụng có thể tham gia vào bất kỳ giai đoạn nào của quy trình

- Do người sử dụng có thể tham gia vào bất kỳ giai đoạn nào của quy trình nên việc phát hiện lỗi và khắc phục sẽ dễ dàng hơn, chi phí ít tốn kém hơn.

- Với mô hình thác nước thì các giai đoạn được định nghĩa, với đầu vào và đầu ra rõ ràng. Mô hình này cơ bản dựa trên tài liệu nhất là trong các giai đoạn đầu, đầu vào và đầu ra đều là tài liệu.Sản phẩm phần mềm được hình thành thông qua chuỗi các hoạt động xây dựng phần mềm theo trình tự rõ ràng

- Người sử dụng không có cơ hội tham gia trong suốt thời gian của các giai đoạn trung gian từ thiết kế cho đến kiểm thử. Đặc biệt với những dự án lớn, người sử dụng chỉ có thể nhận ra rằng hệ thống phần mềm không phù hợp cho nhu cầu của họ vào thời điểm cuối dự án.

- Nói chung, mô hình này thường ẩn chứa nhiều rủi ro mà chỉ có thể phát hiện ở giai đoạn cuối cùng và chi phí để sửa chữa có thể rất cao.

# Mô tả và phân tích luật hình ảnh (gestalt)? (Chương 3 – Slide 7)

* Luật gần gũi: Các mảng được gần lại với nhau được coi là thuộc về nhau.
* Luật đóng cửa: Khu vực bên trong một dây chuyền khép kín được xem là một hình dạng
* Luật tiếp nối tốt: Các mảng trên đường thẳng được coi là thuộc về nhau.
* Luật tương đồng: Những điều mà giống nhau được coi là thuộc về nhau. (trong slide ko có cái này)

# Mô tả và phân tích mô hình ánh xạ? (Chương 4 – Slide 20)

1. **Mô tả :**

Nó là một trong những điều cần biết về tên miền, và một điều nữa đó là làm thế nào các hệ thống máy tính hoạt động được.

- Để làm cho mọi thứ hoạt động trong thực tế, bạn cũng cần phải biết làm thế nào để ánh xạ một thứ đến thứ khác.

- Ví dụ, người dùng phải biết những chức năng hệ thống nào sử dụng để thực hiện một domain task.

+ Ta cần một mô hình tư duy để ánh xạ các nhiệm vụ tới các chức năng.

+ Một giao diện người dùng tốt giúp người dùng thành lập mô hình tư duy đúng cho việc ánh xạ.

1. **Phân tích:** Mô hình tư duy của người dùng cũng nên nói với anh ấy rằng hệ thống máy tính ghi lại những gì được đưa tới và những gì đang trong quá trình, nhưng các bản ghi không phản ánh đầy đủ điều gì sẽ xảy ra trong thực tế.

# Nêu và phân tích 6 nguyên lý cơ bản trong thiết kế giao diện người sử dụng? (Chương 4)

* **Thân thiện với người sử dụng:**
* Giao diện nên dựa vào các thuật ngữ và khái niệm hướng người dùng hơn là các khái niệm máy tính.
* Giao diện nên dùng các thuật ngữ và khái niệm rút ra từ kinh nghiệm của những người sẽ dùng hệ thống nhiều nhất.
* Ví dụ, một hệ thống văn phòng nên dùng các khái niệm như thư từ, tài liệu, thư mục,... hơn là đường dẫn, tên file,...
* **Nhất quán:**
* Hệ thống nên hiển thị một cách nhất quán. Các lệnh và menu nên có cùng định dạng, các dấu chấm lệnh nên tương tự nhau...
* Giao diện cần nhất quán sao cho các thao tác gần giống nhau có thể được kích hoạt theo cùng kiểu.
* **Ít bất ngờ:**
* Nếu một lệnh được thực hiện theo cách thông thường, người dùng có thể dự đoán được thao tác của các lệnh tương tự.
* Người dùng không bao giờ bị bất ngờ về hành vi của hệ thống.
* **Có thể khôi phục được:**
* Hệ thống nên cung cấp một số cơ chế phục hồi lại tình trạng hoạt động bình thường sau khi gặp lỗi. Cơ chế này này có thể bao gồm chức năng undo, xác nhận một hành động hủy ...
* **Hướng dẫn người sử dụng:**
* Một số hướng dẫn người dùng như hệ thống giúp đỡ, tài liệu trực tuyến ... nên được cung cấp.
* Giao diện nên có phản hồi có nghĩa khi xảy ra lỗi và cung cấp các tiện ích trợ giúp theo ngữ cảnh.
* **Đa dạng người sử dụng:**
* Nên cung cấp các tiện ích tương tác cho các loại người dùng khác nhau.
* Ví dụ, một số người dùng có khả năng nhìn hạn chế thì nên để cỡ chữ to hơn.

# Nêu và phân tích sự khác nhau giữa mô tả công việc (task) và vùng làm việc (work area)? (Chương 5 - Slide 5)

* Mô tả công việc (task)
* Task để mô tả 1 **công việc chi tiết** cho người sử dụng.
* Việc mô tả công việc là cần thiết. Vì nó tránh cho người dùng hiểu lầm, thao tác sai dẫn đến giao diện báo lỗi.
* Có nhiều cách để mô tả công việc. Cách mô tả đơn giản nhất là **mô tả công việc theo tên**.

VD: chương trình Quản lý khách sạn:

Công việc Tiếp nhận khách hàng được mô tả theo tên như sau:

Công việc quy trình thuê phòng được mô tả theo tên như sau:

Danh sách công việc (Task list)

T1. Danh sách phòng khách sạn

T2. Đăng ký thuê phòng

T3. Kiểm tra phòng trống

T4. Đổi phòng

T5. Dịch vụ khách sạn (bữa sáng)

* Ngoài cách mô tả công việc theo tên, thì còn phải ghi chú thích cho công việc đó.

VD: Công việc tiếp nhận khách hàng:

T1.Danh sách phòng khách sạn

T1.1. Có thể thuê được nhiều phòng.

T2. Đăng ký thuê phòng

T1.2 Kiểm tra phòng trống. Một số khách hàng đã đặt trước, một số chưa đặt.

T3. Kiểm tra phòng trống

T1.3 Kiểm tra phòng trống. Xem khách hàng đã có thông tin chưa, chưa thì lưu thông tin. Sau đó lập hóa đơn.

T4. Đổi phòng

T1.4 Thay đổi phòng. Có thể thay đổi phòng bất cứ lúc nào trong thời gian thuê phòng.

T5. Dịch vụ khách sạn (bữa sáng)

T1.5 ghi danh sách bữa ăn sáng. Yêu cầu dịch vụ bất cứ lúc nào.

Clever:hiện thị màn hình đặc biệt cho danh sách bữa ăn sáng.

* Vùng làm việc (work area)

- Các mô tả công việc không nói bất cứ điều gì về những **người sử dụng** và các **điều kiện chung** của sản phẩm.

=> Cần: một mô tả riêng của các khu vực làm việc và người dùng chuyên nghiệp.

VD: Khu vực làm việc 1.Tiếp nhận

+Điều kiện: Nhân viên lễ tân tthông thường phải đứng, phải đối mặt với các khách hàng. Nhân viên lễ tân đêm thì thường 1 mình.

+Người sử dụng: Trong ví dụ này gồm 2 loại nhân viên

* nhân viên mới trong các công việc tạm thời, ví dụ trong các kỳ nghỉ hè; nghỉ lễ. Thông tin nhân viên này:
* Không gắn bó lâu dài với công việc
* Thể chất: thị giác bình thường, thính giác, chiều cao,cân nặng, vv
* Kiến thức về CNTT: xử lý văn bản đơn giản. biết các trang Web.
* Thái độ CNTT: Biết 1 phần công việc.
* Kiến thức miền: Biết cơ bản.
* Miền thái độ: Thích công việc.
* nhân viên sử dụng có kinh nghiệm đã trải qua nhiều năm trong việc tiếp nhận, ví dụ các chủ sở hữu. Thông tin nhân viên này:
* Gắn bó với 1 công việc lâu dài.
* Thể chất: thị giác bình thường, thính giác, chiều cao,cân nặng, vv
* Kiến thức về CNTT: xử lý văn bản đơn giản và cần biết nhiều hơn.
* Thái độ CNTT: Tìm hiểu các hoạt động trong công việc.
* Kiến thức miền: Biết tất cả các thủ tục và các trường hợp đặc biệt.
* Miền thái độ: Thích công việc. Muốn trở thành chuyên gia.
* **KẾT LUẬN:**

Một mô tả công việc (task) tốt:

* Được đóng: Sau mối công việc người dùng phải được đóng
* quy tắc Session: nhỏ, liên quan đến "công việc" mà không phá vỡ như một công việc
* Tên miền cấp: Ẩn ai làm gì với ngôn ngữ bắt buộc
* Không chương trình - "nếu khách hàng đã đặt phòng rồi ..."
* Hoàn thành công việc cung cấp cho người dùng một cảm giác dễ chịu.
* Khi thiết kế giao diện người dùng, chúng ta nên quan tâm hỗ trợ toàn bộ công việc - không chỉ mỗi công việc phụ trong sự cô lập.
* Mô tả công việc (task) và vùng làm việc (work area) muốn hỗ trợ hiệu quả cho các nhóm nhiệm vụ chứ không phải chỉ là nhiệm vụ nhỏ.

# Nêu và phân tích sự khác nhau giữa công việc (task) và user case? (Chương 5)

* **Use case:**
* Dùng để đại diện cho các nghiệp vụ của hệ thống. yêu cầu của người
* Sử dụng đối với hệ thống và sự tương tác lẫn nhau giữa
* 1 use case là một công việc nào đó được mô tả rõ ràng,những gì máy tính làm,
* Những gì con người làm để đạt dược mục tiều nào đó
* Use case là phải hoàn tất.
* **Task:**
* Liệt kê chi tiết công việc của người sử dụng phải làm
* Để từ đó hiểu được những mô tả công việc là đúng hay sai

# Nêu và phân tích sự khác nhau giữa cửa sổ ảo (virtual windows) và cửa sổ (windows) trong màn hình thiết kế (screens)? (Chương 6)

* **Ý tưởng cơ bản khi thiết kế một tập hợp các cửa sổ ảo là:**

- Thiết kế vài cửa sổ ảo càng tốt trong khi đảm bảo rằng (1) tất cả các dữ liệu có thể nhìn thấy ở đâu đó và (2) nhiệm vụ quan trọng chỉ cần một vài cửa sổ.

- Đầu tiên chúng ta lập một kế hoạch cho những gì phải ở trong mỗi cửa sổ, và tiếp theo chúng ta làm cho một thiết kế đồ họa chi tiết của cửa sổ.

- Cuối cùng, chúng ta kiểm tra với người dùng rằng họ hiểu các cửa sổ, và chúng ta kiểm tra đối với các mô tả nhiệm vụ và mô hình dữ liệu mà tất cả mọi thứ được đảm bảo.

🡪 Thông thường có rất nhiều thiết kế lặp đi lặp lại ở giai đoạn này.

* Cách chúng ta thiết kế các cửa sổ ảo sẽ giúp chúng ta cân bằng giữa hai thái cực:

- cơ sở dữ liệu theo định hướng và các bước hướng bước theo giao diện người dùng

* Mặc dù chúng ta trình bày các thiết kế dưới đây trong từng bước một, thiết kế không phải là một thủ tục tự động, nơi bạn bắt đầu ở một đầu và kết thúc với một kết quả tốt.
* Một thiết kế tốt luôn luôn bao gồm một số điểm kỳ diệu
* Các bước thủ tục theo từng bước là hỗ trợ tốt cho sự kỳ diệu - không phải là một thay thế cho nó.

**1. Plan the virtual windows (1. Kế hoạch các cửa sổ ảo)**

**Design procedure (Thủ tục thiết kế)**

a) Nhìn vào một nhiệm vụ quan trọng và phổ biến. Hãy tưởng tượng các dữ liệu người dùng muốn thấy để thực hiện nhiệm vụ này.

b) Gom nhóm dữ liệu vào một vài cửa sổ ảo và phác thảo các nội dung của mỗi cửa sổ.

c) Nhìn vào các công việc tiếp theo và tưởng tượng các dữ liệu người dùng muốn thấy để thực hiện nó.

d) Nếu công việc này dụng lại dữ liệu có trong cửa sổ ảo đã lên kế hoạch, xem xét việc tái sử dụng các cửa sổ.

e) Nếu nhiệm vụ sử dụng dữ liệu bổ sung hợp lý liên quan đến một cửa sổ ảo hoạch định, xem xét việc mở rộng cửa sổ này. Nếu không, xác định cửa sổ ảo mới với các dữ liệu bổ sung.

f) Tiếp tục từ bước d với công việc tiếp theo cho đến khi tất cả các nhiệm vụ được hỗ trợ đúng đắn của dữ liệu.

g) Thay đổi: Khi phác thảo những nội dung của một cửa sổ ảo, một số nhà thiết kế chỉ là danh sách các trường để được nhìn thấy, những người khác làm nhiều hơn hoặc ít thành phác thảo đồ họa.

h) Thay đổi: Đối với nhiệm vụ với nhiều bước và nhiều loại dữ liệu, xử lý các bước từng cái một như thể chúng là nhiệm vụ riêng biệt.

i) Thay đổi: Nếu kết quả này có vẻ lộn xộn hoặc quá phức tạp, làm lại kế hoạch, bắt đầu với công việc khác.

**Design rules (Quy tắc thiết kế)**

1. Cửa sổ vài mẫu.

Giữ tổng số cửa sổ mẫu nhỏ. Hãy thử sử dụng lại các mẫu cửa sổ qua nhiệm vụ. (Nó là dễ dàng hơn để nắm bắt vài mẫu cửa sổ như một mô hình về tinh thần.)

2. Rất ít trường hợp cửa sổ cho mỗi nhiệm vụ.

Đối với mỗi công việc, người sử dụng phải truy cập vài trường hợp cửa sổ. (Điều này cải thiện hỗ trợ công việc.)

3. Dữ liệu trong một cửa sổ chỉ dụ.

Tránh người dùng có thể nhìn thấy các mục dữ liệu giống nhau trong một số trường hợp cửa sổ.

4. Bắt nguồn từ một điều.

Một cửa sổ ảo thường sẽ được bắt nguồn từ một đối tượng duy nhất. Nó cho thấy dữ liệu về đối tượng này và các đối tượng liên quan đến các đối tượng.

5 cửa sổ ảo gần kích thước màn hình cuối cùng. Một cửa sổ ảo có thể hơi lớn hơn so với màn hình vật lý có sẵn, nhưng không lớn hơn nhiều. Ở giai đoạn sau của thiết kế, chúng tôi có thể cắt cửa sổ ảo vào một vài màn hình nhỏ hơn hoặc sử dụng thanh cuộn để tạo ra ảo tưởng rằng đây là một màn hình lớn.

6 tổng quan cần thiết của dữ liệu. Các cửa sổ ảo phải cung cấp tổng quan của nhiều dữ liệu, cho dù người dùng có thể cần chỉ là một phần của những dữ liệu này.

8.  Tất cả các dữ liệu truy cập. Thường thì tất cả dữ liệu từ các mô hình dữ liệu phải được nhìn thấy và điều chỉnh được thông qua một số cửa sổ ảo. Nếu không chúng ta có thể thiếu một cửa sổ ảo và có lẽ cũng là một nhiệm vụ.

* **2. Virtual windows, graphical version (2. cửa sổ ảo, phiên bản đồ họa)**

Kế hoạch cửa sổ ảo có thể nhìn khá thuyết phục, và các nhà phát triển có thể kết luận rằng họ có thể đến với việc trình bày đồ họa khi họ thiết kế các màn hình chính thức.

Sau khi họ làm như vậy và cố gắng để lấp đầy một số dữ liệu thực tế, họ nhận ra rằng những phác thảo không hoạt động.

Có thể không có đủ không gian cho dữ liệu thực tế, người dùng không hiểu những gì các cửa sổ hiển thị, hoặc người dùng không có được cái nhìn tổng quan cần thiết.

Kết quả: quay trở lại và thiết kế lại rất nhiều - hoặc bỏ qua khả năng sử dụng.

=> Vì lý do này, điều quan trọng là làm cho một thiết kế đồ họa cẩn thận trong việc kết nối với kế hoạch.

**Design procedure (Thủ tục thiết kế)**

a) Đối với mỗi cửa sổ ảo trong kế hoạch, thực hiện một thiết kế đồ họa chi tiết. Thiết kế chỉ trình bày dữ liệu. (Đừng thêm các nút, menu hoặc các chức năng khác. Còn quá sớm để đấu tranh chống lại những sự phức tạp.)

b) Điền thông tin vào cửa sổ với các dữ liệu thực tế. (Họ có thể cần nhiều không gian hơn bạn mong đợi.)

c) Điền thông tin vào cửa sổ với dữ liệu cực đoan, nhưng thực tế. (Dữ liệu cực cũng phải dễ nhìn thấy.)

d) Nếu cửa sổ không nhìn bên phải, bạn có thể phải thay đổi kế hoạch cho các cửa sổ.

**Sources of graphical design (Các nguồn của thiết kế đồ họa)**

Làm thế nào để bạn thực hiện một thiết kế đồ họa tốt? Làm thế nào để bạn thực hiện một thiết kế đồ họa tốt?

1. Thực hiện theo nền tảng.

Giới hạn mình vào các hình thức trình bày có sẵn trên nền tảng này - trừ khi bạn có lý do rất tốt để làm khác đi. (Nếu không thiết kế sẽ là quá tốn kém cho chương trình sau này.)

2. Nhìn vào các định dạng trình bày dữ liệu thay thế

3. Tái sử dụng thuyết trình dữ liệu cũ - nếu phù hợp.

4. Sử dụng các luật Gestalt làm cho người dùng xem cách mọi thứ liên quan đến nhau.

5. Nghiên cứu thiết kế khác của mọi người.

Hãy tìm những điều tốt trong thiết kế của họ - và những điều xấu trong của riêng bạn.

* **3. Virtual windows in development**

sản phẩm công việc. Một sản phẩm công việc là một thuật ngữ phát triển cho một kết quả trung gian trong quá trình phát triển hệ thống.

a) Các mô hình dữ liệu và mô tả dữ liệu liên quan (dữ liệu từ điển).

b) Các mô tả công việc (trong một trong nhiều hình thức).

c) Các cửa sổ ảo.

d) Một danh sách các lỗi thiết kế hoặc những thứ để tìm hiểu.

* **4. Checking against task descriptions (4. Kiểm tra lại mô tả công việc)**

Khi kết hợp dữ liệu vào cửa sổ ảo, chúng ta đã sử dụng một sự hiểu biết khá trực quan trong những nhiệm vụ và nhu cầu thông tin của họ.

Tuy nhiên, một sự khác biệt giữa các nhà thiết kế tốt và xấu là các nhà thiết kế tốt biết làm thế nào để phát hiện các khiếm khuyết.

# Phương pháp kiểm tra CREDO là gì? Nêu ví dụ minh họa? (Chương 6 - Slide 37)

CREDO là viết tắt của "CREATE, READ, EDIT, DELETE, OVERVIEW" và đại diện cho các hoạt động khác nhau trên dữ liệu trong một hệ thống. Chúng ta có thể kiểm tra xem có tất cả các hoạt động thích hợp trên các dữ liệu trong một hệ thống bằng cách so sánh các cửa sổ giao diện người dùng (cửa sổ ảo tại thời điểm này) đối với các loại dữ liệu khác nhau.

C – Creation of the entity

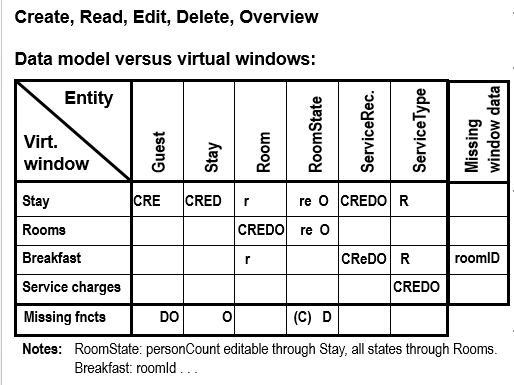
R - Reading of attributes

E - Editing of attributes

D - Deleting the entity

O - Overviewing/searching several entities

VD:



# Tại sao cần hàm điều hướng (navigation) và hàm undo (hoàn tác)? Nêu ví dụ? (chương 7 - Slide 16)

* **Tại sao cần hàm điều hướng (navigation):**

- Giúp chúng ta di chuyển dữ liệu xung quanh trên màn hình, mở cửa sổ hoặc di chuyển đến một trang khác.

- Liên kết trong trang Web.

- Các thanh cuộn để di chuyển xung quanh trong văn bản là một chức năng điều hướng, nhưng cũng có thể khẳng định nó là một chức năng nhập dữ liệu mà thay đổi vị trí tài liệu.

VD: Buttons, siêu liên kết, thanh cuộn, điểm menu, phím cắt ngắn, vv

* **Tại sao cần hàm undo (hoàn tác):**

***- Người sử dụng nhầm lẫn, nhấn phím sai:***

VD: Người dùng muốn kiểm tra một khách đặt phòng, nhưng nhấn Delete thay vì Enter. Kết quả là: Hệ thống hủy đặt phòng. Một nút Undo là cách dễ dàng nếu người dùng nhận ra sai lầm của mình ngay lập tức.

***- Khám phá hệ thống:***

+ Khám phá hệ thống là một cách tốt để tìm hiểu cách sử dụng nó.

+ Tuy nhiên, người dùng thường chọn các chức năng mà không có hiệu quả mong muốn.

+ Các nút Undo là một cách dễ dàng, và nó có thể làm đi làm lại vài lần.

***- Quyết định sai lầm:***

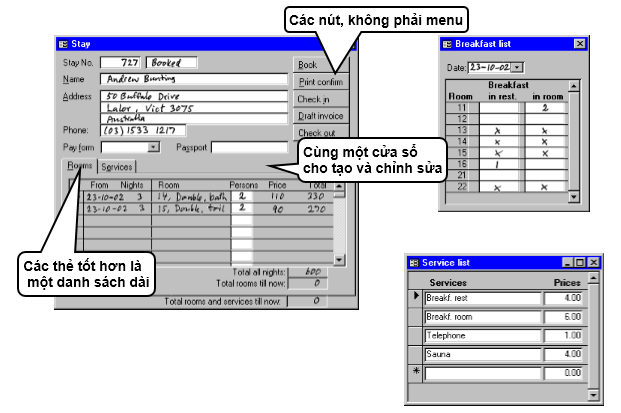
+ Người sử dụng đã làm những điều dự định để làm, nhưng sau đó một thời gian mới nhận ra rằng không nên làm điều đó.

+ VD: Người sử dụng có thể quyết định xóa một tệp từ hệ thống, nhưng mãi sau này nhận ra rằng mình thực sự cần nó một lần nữa.

# Phân biệt giữa khiếm khuyết phần mềm và yêu cầu thay đổi phần mềm, ví dụ và phương án xử lý? (Chương 8 – Slide 13, 14, 24, 25, 31)

|  |  |  |
| --- | --- | --- |
|  | **KHIẾM KHUYẾT PHẦN MỀM**  **(Defects)** | **YÊU CẦU THAY ĐỔI PHẦN MỀM (RFC - Request for change)** |
| **Phân biệt** | Một khiếm khuyết là một vấn đề mà hệ thống không đáp ứng được các yêu cầu đã nêu. | Yêu cầu thay đổi (RFC) là một mong muốn mới của người dùng hoặc khách hàng |

* **Ví dụ: (Slide 13 chương 8)**



* **Phương án xử lý:**

1. **Tạo ra một danh sách hoặc một cơ sở dữ liệu để lưu lại các khiếm khuyết. Nên làm điều đó sớm trong quá trình thiết kế.**

Ghi lại những dữ liệu sau:

+ Xác định lỗi (thường là một số tuần tự)

+ Mô tả các khiếm khuyết

+ Làm thế nào các khiếm khuyết đã được tìm thấy.

+ Việc khắc phục khiếm khuyết - hoặc lý do cho quyết định khác.

+ Tình trạng của các khiếm khuyết

+ Ngày khiếm khuyết đã được tìm thấy

+ Ai tìm thấy nó và đã báo cáo nó

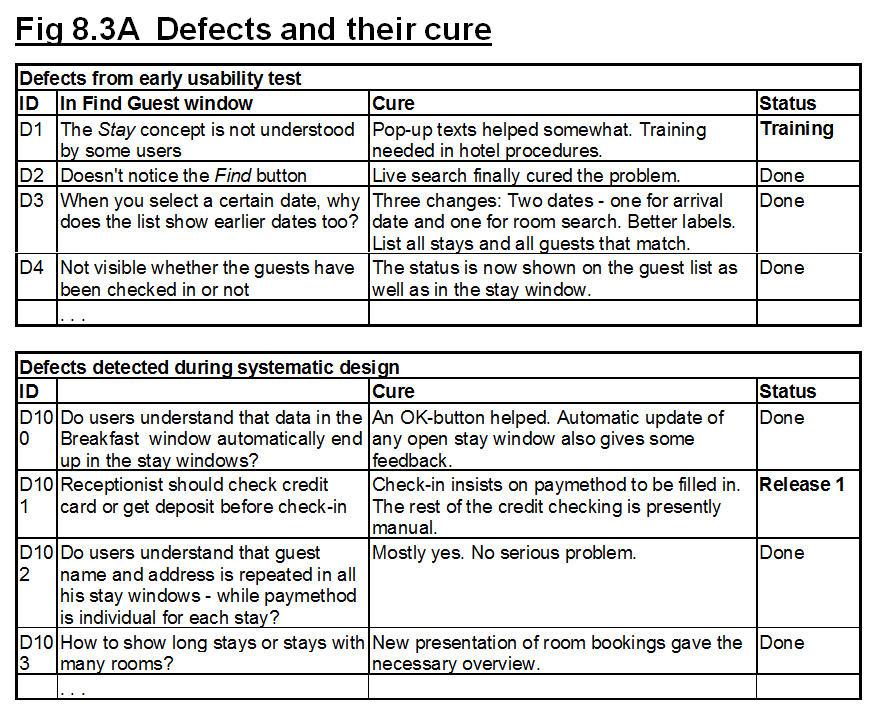
+ Ngày tình trạng đã được thay đổi và do ai.

+ Số giờ làm việc để khắc phục các khiếm khuyết, ví dụ để khắc phục nó.

1. **Ghi lại tất cả các khiếm khuyết (vấn đề) trong quá trình thiết kế, kiểm tra khả năng sử dụng, lập trình và kiểm thử hệ thống.**
2. **Khi một khiếm khuyết được sửa chữa, ghi lại cách chữa và thay đổi trạng thái sang *Done*.**
3. **Định kỳ, rà soát danh sách các khiếm khuyết và quyết định làm gì với những khiếm khuyết chưa được giải quyết. Ghi lại các quyết định, và giải thích các quyết định để khắc phục.**

**VD: Khiếm khuyết trong hệ thống Khách sạn.**

- Hình sau là danh sách đầy đủ của các khiếm khuyết được báo cáo trong quá trình thiết kế của hệ thống khách sạn.



- Hầu hết các vấn đề về tính khả dụng đã được phát hiện trong quá trình kiểm tra khả năng sử dụng của nguyên mẫu hệ thống và đã được gỡ bỏ ngay lập tức, không bao giờ đưa vào danh sách khiếm khuyết chính thức.

- Danh sách này được chia thành nhiều phần theo cách các khiếm khuyết đã được tìm thấy.

+ Kiểm tra tính khả dụng ban đầu của bản vẽ tay mock-up.

+ Trong suốt quá trình thiết kế và kiểm tra quy trình hệ thống.