**12.3 USER INTERFACES: CÁC YÊU CẦU CHI TIẾT**

Nhớ lại các bước dưới đây trong các user interfaces cụ thể

**Step 1**: hiểu biết về user

**Step 2**: hiểu chức năng nghiệp vụ trong quá trình hỏi user

**Step 3:** ứng dụng các nguyên lý về thiết kế màn hình tốt

**Step 4:** lựa chọn các loại cửa sổ (windows) phù hợp

**Step 5:** phát triển menu cho hệ thống

**Step 6:** chọn các controls thiết bị phù hợp

**Step 7:** chọn ra các controls màn hình

**Step 8:** tổ chức các lay out windows

**Step 9:** chọn tông màu phù hợp

**Step 10:** tạo các icons rõ nghĩa

**Step 11:** cung cấp các message hiệu quả, mang tính feedback và thể hiện được sự hướng dẫn tới user

Đảm bảo tính nhất quán giữa các màn hình của các ứng dụng được thiết kế, giữa các màn hình với nhau

+ các chuẩn phong cách, các thủ tục, có khả năng nhìn dễ hiểu và cảm nhận được, vị trí phù hợp

Dự đoán user thường sẽ start tại đâu

+ thường là phía bên trái

Tạo ra các điều hướng càng đơn giản càng tốt

+ căng chỉnh các phần tử / control

+ nhóm các phần tử / control có liên quan với nhau

+ xem xét các viền bao quanh các phần tử / control

Ứng dụng một kế thừa để làm nổi bật thứ tự quan trọng

Ứng dụng các nguyên lý làm trực quan:

+ cân bằng, đối xứng, chuyên nghiệp, khả năng dự đoán

+ đơn giản, hữu dụng, cân đối, tiết kiệm không gian

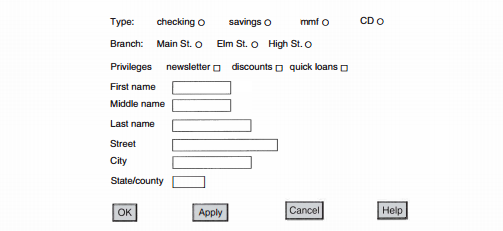
Cung cấp các tiêu đề

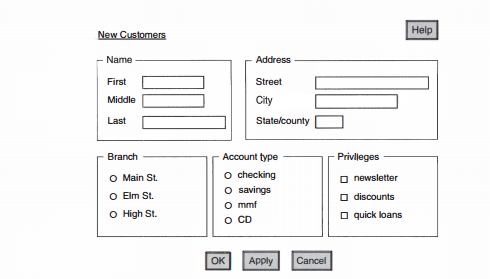
Các bước 1 và 2 được mô tả ở chương 11, vì nó được ứng dụng chủ yếu cho các yêu cầu cấp cao. Chúng ta mô tả các bước còn lại cho các yêu cầu về giao diện chi tiết, cụ thể

**Bước 3:** ứng dụng các nguyên lý của thiết kế màn hình tốt

Hình dưới đây liệt kê các thành phần chính để thiết kế một màn hình tốt. Trong hình bao gồm các hệ số thường dùng để tạo ra sự hài lòng về giao diện. Mặc dù các thành phần đó chỉ dùng để giới thiệu các vấn đề về hiệu quả trực quan, tuy nhiên các thành phần đó có thể sử dụng bởi nền công nghệ phần mềm hạng trung

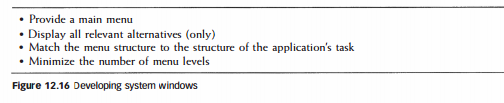
Ví dụ, chúng ta sử dụng một vài nguyên lý cho một màn hình được sử dụng để nhập thông tin về khách hàng và tài khoản của họ. Một thiết kế tạm được khởi tạo tại tầng GUI được show dưới đây. Để cải thiện interface, chúng ta bắt đầu tại top-left, đặt thành phần / control có quan trọng nhất, và nhóm các phần tử giống nhau.





**Step 5:** phát triển các menu hệ thống

Một vài quy tắc để tạo ra các main menu, được cung cấp bởi Calitz [3], được show dưới đây



Các user yêu cầu khả năng anchor (neo) ổn định, toàn vẹn cho ứng dụng, do đó cần các main menu cố định, ko thay đổi. Số lượng các items trên menu thường nên nằm giữa 5 và 9, vì hầu hết chúng ta cảm thấy thoải mái với lựa chọn đó. Chẳng hạn, bộ xử lý word với cuốn sách này chỉ có 9 main menu item: File, Eedit, View, Insert, Format, Tools, Table, Window, Help. Số lượng item có thể nhiều hơn, vì có nhiều không gian trống. Tuy nhiên, chúng ta có lẻ tìm kiếm liên tục danh sách các option mà chúng ta yêu cầu, và các kết quả ngoài mong đợi làm gia tăng sự lựa chọn.

Các items trong main menu được xác định bởi nghiệp vụ, trong tình huống này, ta cần xử lý về câu chữ. Ví dụ, các lệnh vè đồ họa sẽ được đặt trong menu thứ cấp

**Step 6:** lên danh sách các controls thiết bị

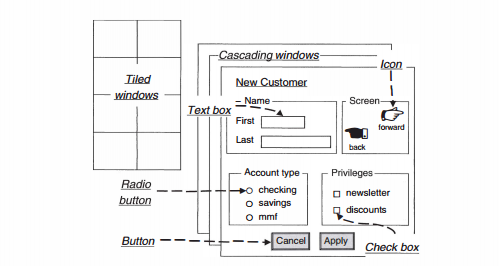
“Các controls thiết bị” là nghĩa vật lý bởi user communicate các mong muốn của họ tới ứng dụng, bao gòm joysticks, trackballls, máy tính bảng đồ họa, các bút màn hình, chuột, microphones and keyboards.

**Step 7:** Chọn ra các controls màn hình

“Controls màn hình” là các biểu tượng xuất hiện trên màn hình, user sẽ chú ý đến phần input của ứng dụng, bao gồm icons, buttons, text boxes, selections, and radio buttons. Các quy tắc sắp xếp các controls trên một cửa sổ giống với thiết kế màn.

**Step 8:** tổ chức layout windows.

Các quy tức lên lay out cho nhiều cửa sổ tương tự như thiết kế từng màn hình của chúng, bao gồm có sự đối xứng, cân bằng, … Nhưng chúng có liên quan đến sắp xếp và phân tầng, được minh họa trong hình dưới.



**Step 9:** Chọn màu phù hợp

Color không tự động làm tạo ra một user interface hữu dụng hơn hoặc hấp dẫn hơn, tuy nhiên, nó có thể dễ dàng làm xấu đi thiết kế. Theo nhà thiết kế danh tiếng Paul Rand, “mầu sắc là tính cách phức tạp”. Các kỹ sư phần mềm không phải là những nhà thiết kế chuyên nghiệp. Luôn sử dụng màu đen hoặc màu trắng trước tiên. Nếu nó làm rõ được mục đích, hãy giới thiệu mọt màu khác. Hãy đảm bảo nó có thể giúp ích được user. Suy nghĩ một cách nghiêm túc trước khi thêm vào một color.

Note lại các ứng dụng được làm tốt, chẳng hạn bộ sử lý word được sử dụng rộng rãi, có thể gợi ý các màu tốt để sử dụng. Bạn có thể được bảo đảm cho các giao diện được thiết kế chuyên nghiệp cho trải nghiệm, các kỹ sư phần mềm chưa thạo có thể được lợi từ điều này.

**12.4 Các yêu cầu bảo mật chi tiết**

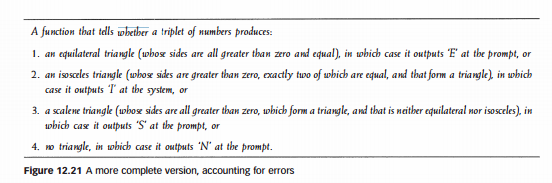
Viết các yêu cầu cụ thể cho các biện pháp bảo mật là một vấn đề hoàn toàn cụ thể và cấp bách.

Một vài trong số đó được thể hiện ở hình 12.18 và 12.19. Khi thoát khỏi bảo mật, ví dụ, chúng ta có muốn yêu cầu rằng liệu không có keystroke trên ứng dụng trong khoảng 10 phút, nó sẽ tự động đăng xuất. Kiểm tra các lỗ hỗng bảo mật là cực kỳ quan trọng. Một vài yêu cầu dễ dàng đủ để xác định, nhưng khó mà thực thi. Chẳng hạn, làm thế nào một ứng dụng biết có ai đó đã phá vỡ bảo mật.

**12. 5 Các điều kiện lỗi**

Mỗi yêu cầu, chúng ta hỏi chuyện gì xảy ra nếu nó bị lỗi. Ví dụ, làm một yêu cầu như hình 12.20: một hàm nhận 3 số để tạo một tam giác đều, một tam giác cân, hoặc một tam giác thường

Đặc tả yêu cầu này không thể hoàn thành được vì nó không quản lý hết các lôi về điều kiên. Phiên bản ở hình 12.12 thì đầy đủ hơn.



Dịch: Một hàm nhận 3 số, liệu có thể là vẽ ra được:

1. Một tam giác đều, có 3 cạnh bằng nhau và lớn hơn 0, và xuất ra màn hình “E”
2. Một tam giác cân, có 3 cạnh bằng nhau và lớn hơn 0 và chỉ có 2 cạnh bằng bằng nhau, và xuất ra màn hình “I”.
3. Một tam giác thường, có 3 cạnh lớn hơn 0, và không có cạnh naoof bằng nhau, xuất ra màn hình “S”.
4. Không có một tam giác nào, xuất ra màn hình “N”.

Thiếu điều kiện lỗi trong đặc tả yêu cầu dẫn đến function cần phải được test, vì tester tập trung vào điều kiện lỗi, và cần biết yêu cầu về output.

Phân tích yêu cầu không trả về input bất hợp lệ: nó giải quyết với chúng một cách trực tiếp. Chẳng hạn, nó đang tạm thời để giả sử rằng GUI cho yêu cầu về tam giác không cho phép input số âm, và vì thế function không phải giải quyết với lỗi trên. Tăng sự phụ thuộc vào các phần ứng dụng, những nơi chúng ta có thể đạt được các phần độc lập. Mặc dù nó là một thói quen tốt để bẫy các lỗi user input không hợp lệ tại tầng GUI và để đảm bảo user chỉ nhập vào những giá trị hợp lệ, điều này không thay để cho một đặt tả tốt và nhận dạng lỗi. Các tác giả khuyến nghị yêu cầu các bẫy về lỗi dữ liệu từ input tại một vài điểm, không phải tất cả. Điều này tương đương với thói quen kỹ sư được thành lập từ từ.