Non functional requirement

Tags

Các loại requirement trong một dự án phần mềm

Như anh em đã biết, hoặc chưa biết: *một giải pháp, một sản phẩm, hay một phần mềm nào đó đều có các yêu cầu cụ thể (requirement) cho các giải pháp, sản phẩm hay phần mềm đó.*

Là một người làm Business Analyst, chúng ta sẽ làm rất nhiều thứ xoay quanh các requirement này.

Requirement thì có 4 loại:

- Business Requirement
- Stakeholder Requirement
- Solution Requirement
- Transition Requirement.

Bất kỳ phần mềm nào cũng vậy? Sinh ra đều phải có mục đích. Tức mỗi phần mềm đều có các yêu cầu của riêng nó. Mà các yêu cầu này không phải là ít.

Một phần mềm có rất-rất-rất nhiều yêu cầu. Nào là phải làm được cái này, cái kia, nào là phải đẹp, phải nhanh, phải abc, xyz...

Chính vì có quá nhiều requirement, xuất phát từ nhiều đối tượng khác nhau. Lộn xộn quá, nên người ta mới gom nó lại, rồi chia thành 4 loại requirement như trên, để anh em BA chúng ta có thể dễ dàng moi móc và quản lý được nó.

Cụ thể 4 loại nó như thế nào thì mình sẽ để dành nói ở bài sau. Bài này mình sẽ tập trung nói về Solution Requirement.

Ô kê, Solution Requirement được chia nhỏ thành 2 loại sau:

- **Functional requirement**
- và **Non-Functional requirement.**

Có một số ví dụ cho anh em dễ hình dung hơn:

Ly nước:

- *Functional Req: ly đựng được nước*
- *Non-Functional Req: ly rớt không bể.*

Mũ bảo hiểm:

- *Functonal Req: có đèn chiếu sáng, nhấp nháy lòe loẹt trong đêm*
- *Non-Functional Req: chịu được lực va đập lên tới 3000 Newton.*

Ly chè đậu đen của bà Bảy đầu xóm:

- *Functional Req: chóng đói, bổ sung vitamin đậu đen.*
- *Non-Functional Req: ăn xong có khăn giấy lau miệng, hoặc ăn xong không đau bụng.*

Phần mềm ABCDXYZ:

- *Functional Req: quản lý thông tin khách hàng*
- *Non-Functional Req: có nút Help – hướng dẫn người dùng online ngay trên hệ thống.*

Non-Functional Requirement là gì?

BABOK ver3.0 định nghĩa Non-Functional Requirement như sau:

> Not relate directly to the behaviour of functionality of the solution, but rather describe conditions under which a solution must remain effective or qualities that a solution must have.
>
> BABOK ver3.0

Lại dài loằng ngoằng, nhưng không sao, đọc cũng dễ hiểu. Càng về sau BABOK định nghĩa càng dễ hiểu mà, hehe.

Non-Functional Requirement là những thứ:

- Không liên quan trực tiếp tới hành vi – chức năng của giải pháp
- Nhưng lại là các điều kiện giúp hệ thống chạy tốt và đảm bảo được chất lượng như yêu cầu.

Rút gọn 2 dòng trên, mình có cách định nghĩa trực quan hơn như sau:

> Non-Functional Requirement = Quality of Services

Tức Non-Functional Requirement là những thứ liên quan đến CHẤT LƯỢNG sản phẩm.

Sản phẩm đó đáp ứng được mục đích sử dụng là 1 chuyện, nhưng nó phải đảm bảo tốt về mặt trải nghiệm người dùng thì mới thật sự đẳng cấp 😎

Important

Quay lại cái xô ở đầu bài. **Xô**, là giải pháp để đựng nước và được dùng cho các mục đích khác nhau.

- ***Functional Req*** *của cái Xô là đựng được nước. Chỉ nhiêu đó thôi.*
- *Còn* ***Non-Functional Req*** *của cái Xô là: xô làm bằng nhựa không giòn, phơi nắng không bể, không mọc rêu, không trơn, kiểu dáng thon gọn, vừa tay cầm.*

Rõ ràng, cái xô ở nhà mình chỉ đạt được mỗi Functional Requirement, còn Non-Functional Requirement thì.. thấy gớm.

Mẹ mình thích dùng vì mẹ mình **chỉ quan tâm** đến Functional Requirement, tức là chức năng của nó, chỉ cần đựng được nước là được.

Còn mình không thích dùng vì nó không đáp ứng được Non-Functional Requirement, những requirement mà đáng lý ra một cái xô phải có. Đã zậy, còn lủng lỗ, chắp vá tùm lum tùm la, thấy ớn.

Type of non-functional requirement

Security

NFR 1: Security (Nguồn ảnh: SLANE Cartoons)

Security là các yêu cầu về **bảo mật** trong quá trình sử dụng hệ thống. Từ lúc truy cập, thực hiện thao tác, in ấn chứng từ, đến lúc đăng xuất khỏi hệ thống.

Ví dụ về Security:

- *Password của người dùng phải được hash bằng MD5.*
- *Hệ thống sẽ deactivate 30 phút nếu người dùng nhập password sai 5 lần liên tiếp.*
- *Tất cả những data "nhạy cảm" của người dùng như: password, SĐT, CMND, email phải được mã hóa bằng 1024bit SSL.*
- *Khi user quên mật khẩu, link tạo mật khẩu mới phải được gửi về duy nhất địa chỉ email đăng ký đầu tiên.*
- *Hoặc khi user thực hiện thanh toán online, hệ thống không được phép lưu trữ thông tin thẻ credit/ debit của user.*

Performance

Phần này có lẽ quá rõ ràng với anh em rồi chứ hả 😎 Performance tức là **hiệu suất hoạt động** của hệ thống, và thường được đo lường bằng những tiêu chí sau:

- Thời gian phản hồi cho một transaction
- Số lượng transaction thực hiện được trong mỗi giây
- Công suất (capacity), ví dụ số lượng transaction mà hệ thống có thể lưu trữ/ thực hiện cho mỗi đối tượng.
- Hoặc các yếu tố về tài nguyên sử dụng như: RAM, dung lượng DB...

Một số ví dụ để anh em hình dung rõ hơn về Performance khi lấy yêu cầu từ khách hàng*:*

*Tất cả những màn hình input và output dữ liệu cần phải được sẵn sàng để hiển thị cho người dùng **trong vòng 3 giây**, với điều kiện tải và connection giữa cilent/server là bình thường, cụ thể:*

- ***Đối với màn hình input:** tối đa 30 trường dữ liệu, không tính toán dữ liệu phức tạp, không tương tác với hệ thống ngoài, có thể lưu trữ dữ liệu trực tiếp ngay xuống DB, và không lưu trữ các tệp nội dung lớn như: hình ảnh, video, tệp tin quá 3MB.*
- ***Đối với màn hình output:** dữ liệu được query trực tiếp từ DB, hạn chế những query phức tạp, những query từ hệ thống ngoài.Hiển thị tối đa 50 dòng dữ liệu, mỗi dòng tối đa 10 cột, và mỗi dữ liệu có độ dài nhỏ hơn 100 ký tự.*
- ***Điều kiện tải bình thường:** 30 CCU (concurrent user – người dùng đồng thời) khi không dùng cân bằng tải.*
- ***Điều kiện server tối thiểu:** Intel Core i5, 4GB RAM, 500GB hard disk.*
- ***Client/ Server Connection:*** 500KB/s

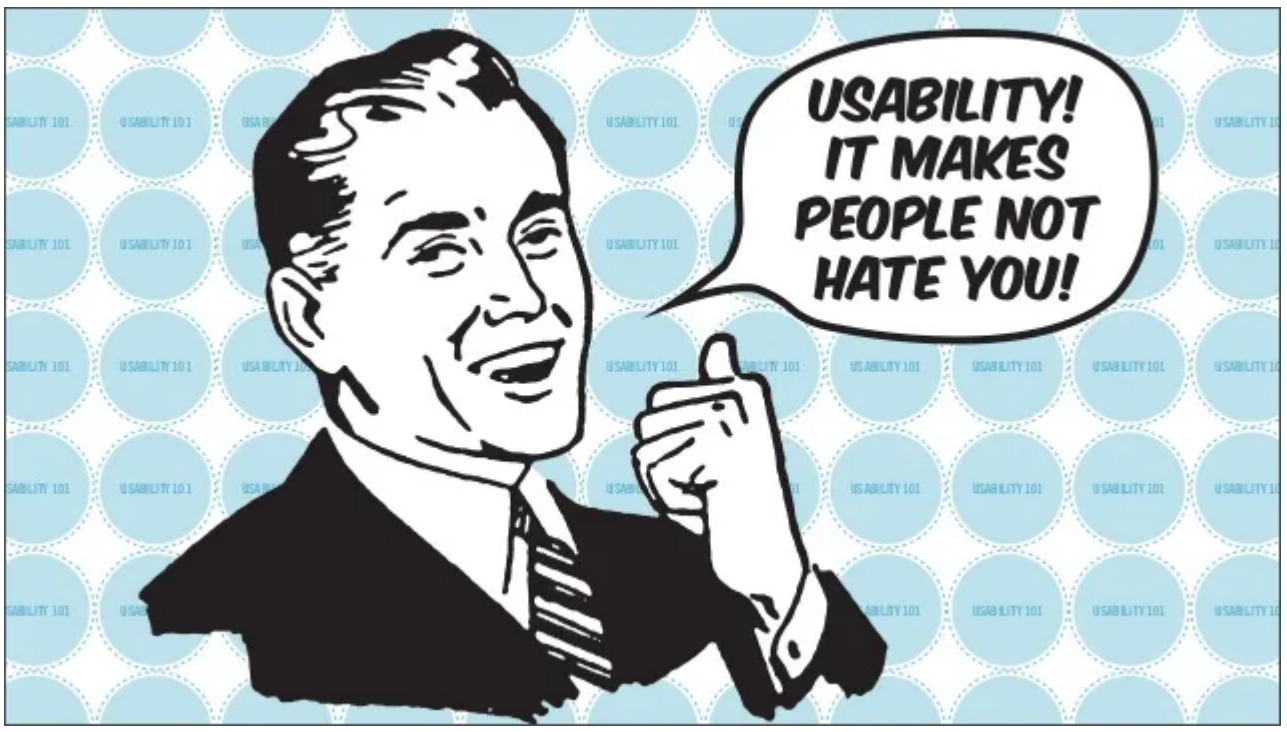
Đối với những anh em làm software development, những điều trên là tối quan trọng. Nhưng đối với anh em làm triển khai, thì những điều trên có vẻ hơi... thừa.

Vì hãng đã lo những thứ này cho mình hết, mình chỉ giải thích cho khách hàng hiểu để tránh yêu cầu mở rộng sau này.

Tuy nhiên, cũng không nên lơ là mà bỏ nó ra khỏi document. Mình cũng đã từng như vậy. Hệ thống khi đó phải query tới rất nhiều những component bên ngoài cho từng transaction một.

Và dĩ nhiên, performance bị kéo tụt hẳn xuống dưới đáy và mình bị complain rất nhiều về vấn đề này. *Lỗi do ai? Chính là do mình,* ***do BA không làm rõ những vấn đề này ngay từ đầu.*** Don't be like me!

3. Usability



Làm rõ các yêu cầu về Usability là sẽ giúp user happy hơn rất nhiều 🙂

Là một trong những NFR quan trọng bậc nhất, Usability chính là khả năng **"dễ sử dụng"** của hệ thống.

Nói về tính *"dễ sử dụng"* thì có 5 yếu tố sau, anh em có thể dựa vào đó để **xác nhận yêu cầu** với khách hàng.

(Xác nhận chứ không phải lấy yêu cầu, vì thường khách hàng họ cũng chả mấy khi để ý đến vấn đề này, trừ khi anh em đang làm product với end user cũng chính là end consumer).

a) Effectiveness

Là tính hiệu quả, **làm được đúng** những gì **kỳ vọng**.

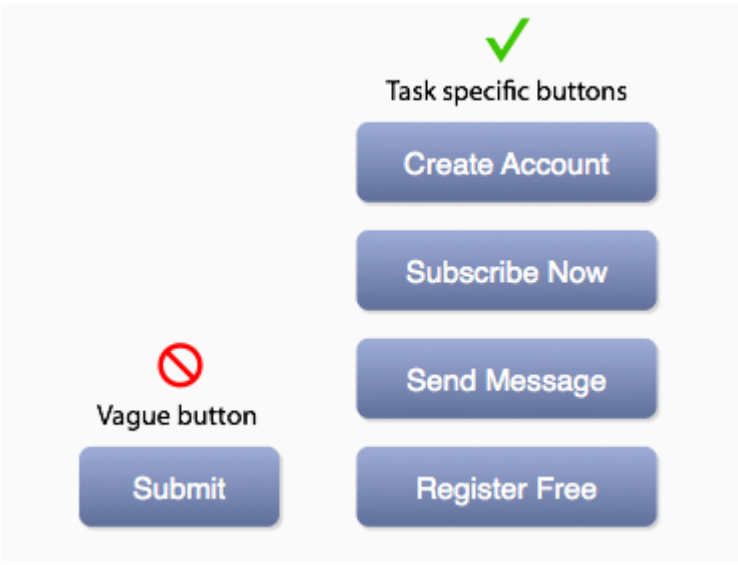
Ví dụ user muốn xem hàng tồn kho thì thay vì user chọn menu On-hand >> rồi chọn kho để xem, thì hãy hiển thị số lượng tồn kho ngay bên cạnh tên kho để user có thể xem được ngay, mà không cần nhấp thêm phát nữa.

b) Efficiency

Cũng là tính hiệu quả, nhưng ngụ ý nhanh hơn, nguy hiểm hơn, thông minh hơn; nói chung là **nhanh và chính xác** hơn.

Ví dụ BA chúng ta phải thiết kế làm sao để: ***giảm thiểu tối đa thao tác*** *của users trên màn hình để họ hoàn thành 1 task trên hệ thống nhanh nhất có thể.*

Ví dụ từ 5 chạm, giảm xuống còn 3 chạm là có thể order xong ly trà sữa. Hoặc nút bấm có label rõ ràng để user hiểu ngay được hành động sau khi bấm nút.



Nguồn ảnh: uxmovement.com

c) Engagement

Là mức độ: **UI của hệ thống *"tương tác"* với khách hàng** như thế nào. Không chỉ đẹp, mà còn phải phù hợp với đối tượng end user và ngữ cảnh sử dụng.

Ví dụ hệ thống quản lý sản xuất thì không thể nào design trông như concưng.com được.

Nó tạo ra chút cảm xúc hơi hơi cu te chút xíu, mà những người làm planning trong sản xuất trên dưới 30 tuổi không hề mong đợi một hệ thống như vậy.

d) Error Tolerance

Tolerance là mức độ *"dễ tha thứ"* của hệ thống. Nghe thấy hài quá, để ví dụ cho anh em dễ hình dung.

Ví dụ khi tạo khách hàng, user nhập 20 trường thông tin, rồi nhấn nút Save. Khi đó, hệ thống tiến hành kiểm tra và phát hiện có 3/20 trường dữ liệu bị nhập sai kiểu dữ liệu.

Lúc này hệ thống thấy ghét quá, chơi xóa hết cha nó 20 dữ liệu mà user đã nhập, yêu cầu user nhập lại từ đầu, zậy mới ác chứ.

Đó là khi Error Toletance của hệ thống cực kỳ thấp.

Thay vào đó, hãy kiểm tra kiểu kiểu dữ liệu ngay trên từng field, và cảnh báo ngay nếu có lỗi, đừng bắt user nhập đi nhập lại nhiều lần.

e) Easy to learn

Là tính *"dễ học"* của hệ thống. Thường người ta sẽ nhìn vào độ consistent (độ nhất quán) của các form màn hình để đánh giá.

Ví dụ ở form màn hình Khách hàng, thanh Navigation ở vị trí A, thanh Command ở vị trí B, nút NEW ở vị trí C. Mà qua form màn hình Đơn hàng, thanh Navigation thì ở vị trí C, thanh Command ở vị trí A, nút New ở vị trí B, là coi như xong phim user luôn.

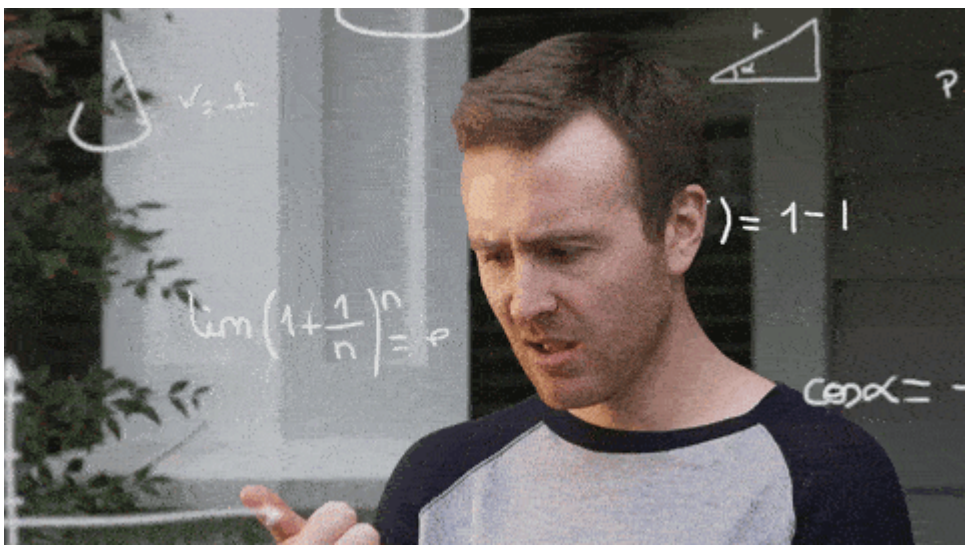
Họ không thể nhớ được, và cũng chả cần phải nhớ.

Phần nhiều các NFR thuộc nhóm Usability là mang kiến thức của UX, anh em có thể tham khảo thêm về Usability tại đây nhé.

4. Integrity

Mục này là nói về *"độ chính trực"* của dữ liệu. Tức độ chính xác, xác thực của dữ liệu.

Ví dụ từ dữ liệu A được input vào hệ thống, nó xử lý hầm bà lằng ra kết quả A′, A″, rồi sang B, C, D, D′, D″ rồi ra E, cộng với F, chia cho G rồi tham chiếu với H để ra được một kết quả J cuối cùng.



Trong quá trình xử lý dữ liệu A để ra được dữ liệu "J" cuối cùng, hệ thống sẽ có các bộ batch jobs chạy ngầm bên dưới, chạy đồng bộ và cả bất đồng bộ.

Nói về integrity, khách hàng sẽ yêu cầu dữ liệu phải được tính toán **chính xác tại-tất-cả-các-thời-điểm**. Chứ không phải lúc này dữ liệu ra chính xác, lúc kia dữ liệu ra sai.

Nói thì nghe mắc cười quá, vì máy tính chứ có phải con người tính đâu mà lúc đúng lúc sai.

Nhưng thực tế đối vối những công thức phức tạp, đòi hỏi lấy dữ liệu từ nhiều nguồn thì chuyện này là có xảy ra chứ không phải không.

Như cá nhân mình đã từng gặp một trường hợp: *để ra được con số thành tiền cuối cùng, hệ thống phải lấy toàn bộ thông tin đơn hàng, request đến dữ liệu khuyến mãi bên ngoài hệ thống để lấy giá, discount, hàng tặng các kiểu, rồi tính toán thêm 4-5 parameters gì nữa để ra được giá cuối cùng.*

Trong quá trình tính toán này có các batch job chạy bất đồng bộ, đòi hỏi 1 đoạn thời gian khoảng 30s đến 1 phút mới chạy xong hoặc người dùng phải chịu khó bấm nút force các job đó thì nó mới chạy.

Khi những job này không chạy thì vô tình những batch job chạy đồng bộ khác nó overlap lên, thế là kết quả bị thiếu đi 1-2 phép tính gì đó, nên kết quả lúc ra sai, lúc ra đúng.

Ra kết quả sai là khi user refresh màn hình ngay, ra kết quả đúng là khi user chờ khoảng 30s rồi mới refresh màn hình hoặc chờ cho hệ thống tự refresh.

Lúc này thì thôi rồi, chỉ có đội quần chứ nói năng gì nữa. Hệ thống làm gì mà lúc ra sai, lúc ra đúng. Từ đó ***nó tạo ra tâm lý ngờ vực rất lớn của khách hàng về giải pháp mình mang lại.***

Do đó, anh em cần phải thống nhất rõ với khách hàng về độ *"integrity"* – độ *"chính trực"* của dữ liệu trong những bối cảnh, những khoảng thời gian nhất định.

Nhớ integrity nhé anh em!

5. Availability

Availability là các yêu cầu về: **mức độ hệ thống sẵn sàng** để được sử dụng, gồm 2 yếu tố:

- Thời gian có thể sử dụng hệ thống
- Các yếu tố phụ thuộc để vận hành hệ thống.

Ví dụ về thời gian nhé: *Hệ thống phải đảm bảo vận hành 24/7, nâng cấp tối đa 1 lần trong vòng 3 tháng, downtime mỗi năm không quá 1 giờ đồng hồ.*

Còn về các yếu tố phụ thuộc thì mình cũng bị nhiều trường hợp.

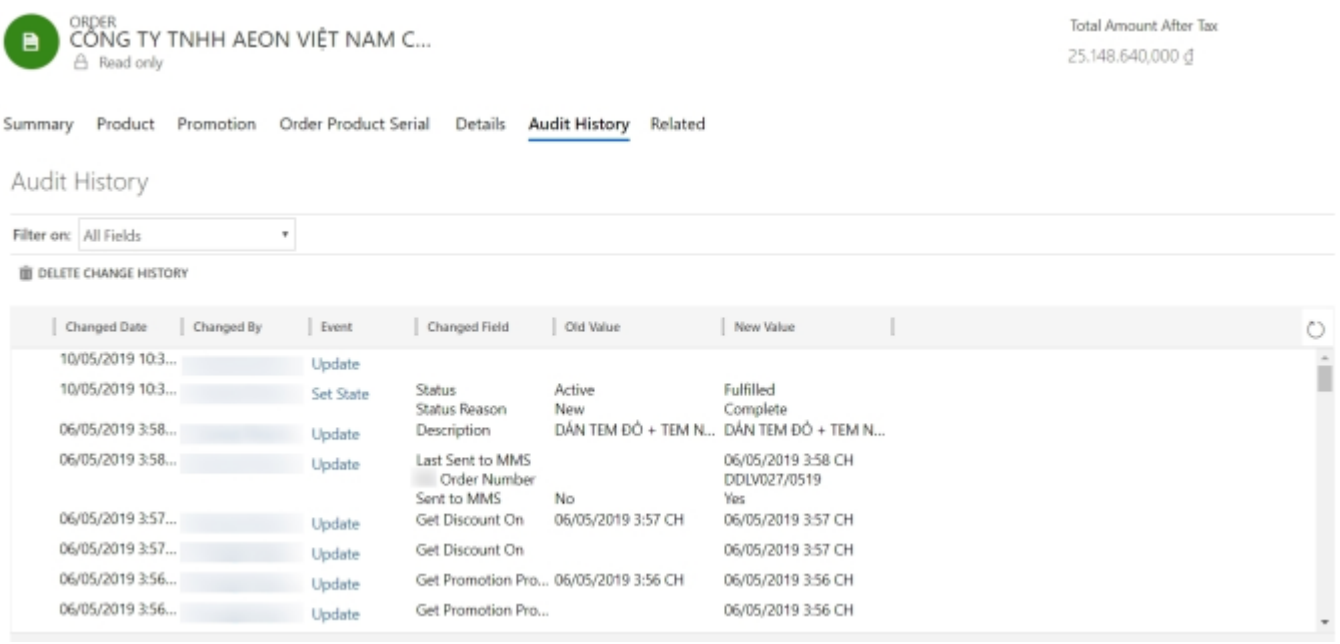
Ví dụ hệ thống của mình đang cài thêm một *"Add-on"* hoặc một ứng dụng nào được built bởi bên thứ ba, thì NFR ở đây có thể là: *Hệ thống phải đảm bảo vận hành 24/7 và không phụ thuộc vào sự ngưng hoạt động của bất kỳ sản phẩm đến từ bên thứ 3 nào.*

6. Audit

Audit là các yêu cầu về **khả năng ghi nhận lại các tác vụ đã thực hiện** trên hệ thống, nhằm mục đích kiểm tra. Nhớ nhé anh em, chỉ nhằm *mục đích kiểm tra*, không phải để thống kê hay báo cáo.

Ví dụ về Audit:

- *Dữ liệu audit được sẽ lưu trữ tách biệt trong một database riêng, khác database chính của hệ thống.*
- *Dữ liệu hệ thống phải được backup hằng ngày, và tồn tại trong vòng 30 ngày.*
- *Các dữ liệu audit được phải ở chế độ Read Only và không được sửa từ giao diện người dùng.*



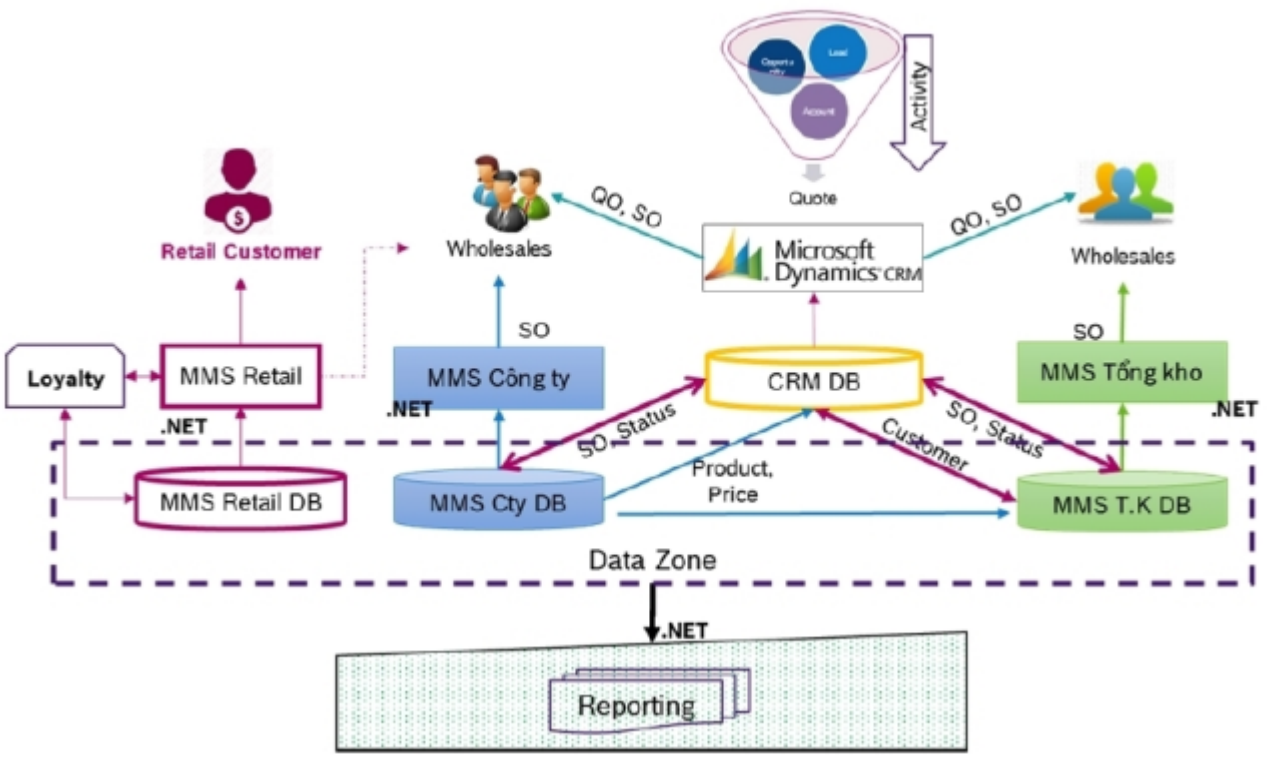
7. External Interface

Phần này nôm na nghĩa là **tích hợp**.

Anh em sẽ elicit các yêu cầu cụ thể của khách hàng về vấn đề: tích hợp các components ở hệ thống đang build với các components ở các hệ thống khác như thế nào.

Vì chuyện tích hợp (integration) là cực kỳ quan trọng trong bất kỳ dự án nào, nên thường mình thấy phần mô tả requirement về tích hợp sẽ được tách riêng ra một phần riêng biệt trong document.

Phần này anh em sẽ mô tả một bức tranh tổng quan cho việc trao đổi dữ liệu giữa các system, bao gồm: *những component nào được trao đổi, mục đích làm gì, tần suất thay đổi ra sao, số lượng người dùng tác động...*



Anh em sẽ phối hợp với Tech Lead/ PM để mô hình hóa lại bức tranh tích hợp dữ liệu của khách hàng

Tuy nhiên ở phần External Interface, anh em cũng có thể thêm 1 vài yêu cầu chi tiết mà mình elicit được từ phía khách hàng như:

- *Toàn bộ dữ liệu phải được tích hợp qua API*
- *Và các API phải trả kết quả ra dạng JSON hoặc XML*
- *Hoặc toàn bộ quá trình nhận/ gửi dữ liệu giữa các hệ thống phải được mô tả chi tiết trong document.*

8. User Interface

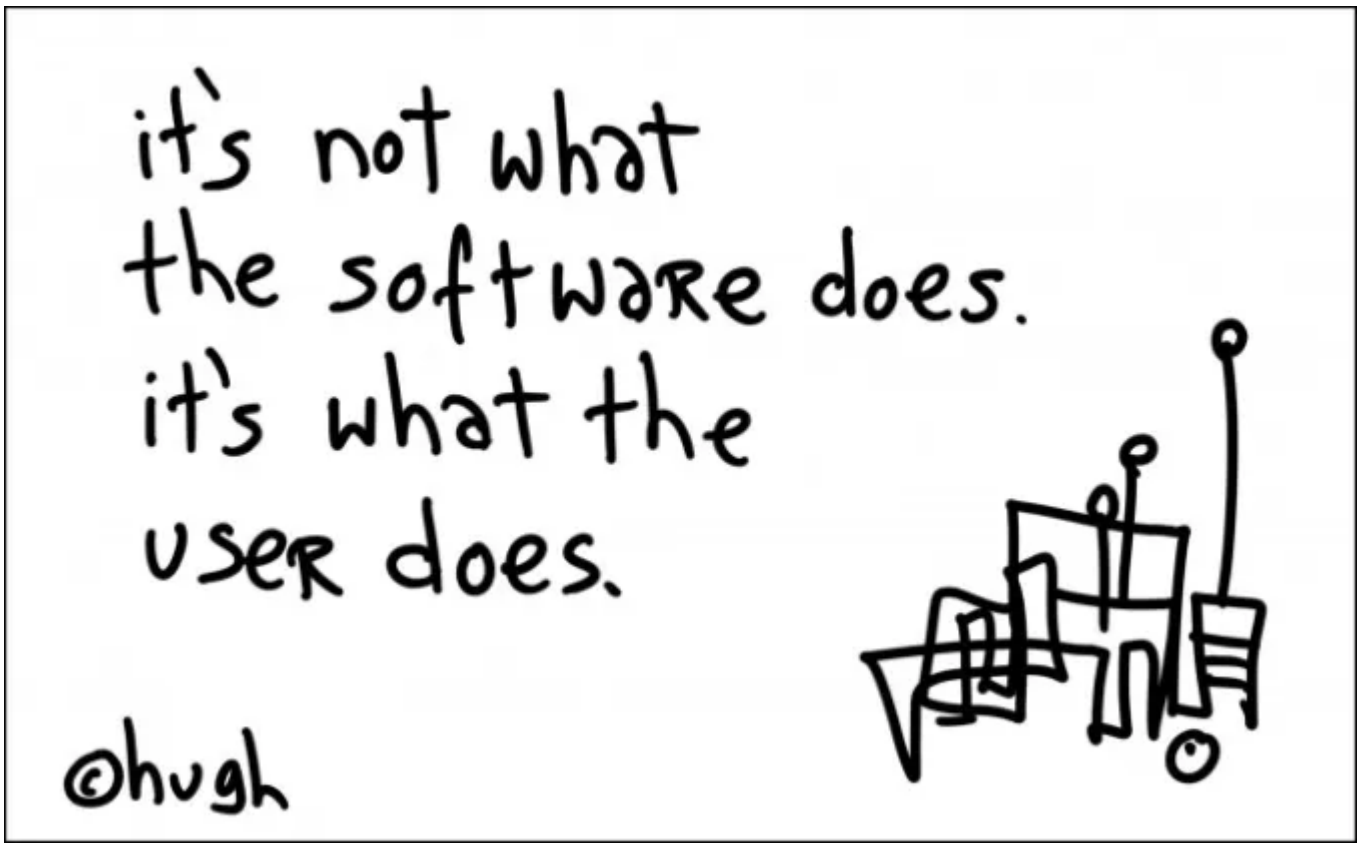
Có thể anh em sẽ dễ nhầm lẫn giữa NFR Usability và User Interface. Nó đều đa phần thể hiện ở giao diện người dùng. Nhưng:

- ***Usability** mang nghĩa **rộng hơn về tính "dễ sử dụng"**, nó có thể là cải thiện UI, thay đổi cấu trúc dữ liệu, hoặc thậm chí là thay đổi luồng data đi trong hệ thống.*
- *Còn **User Interface** chỉ đơn thuần là những **yêu cầu về giao diện người dùng** mà Khách hàng mong muốn, bao gồm:*

- *Cấu trúc UI*
- *Cảnh báo và thông báo lỗi*
- Một số yêu cầu khác.

Một số ví dụ về User Interface như sau:

- *Các lưới dữ liệu xuất hiện trên hệ thống đều phải có chức năng filter và sort.*
- *Hệ thống đều phải hỏi xác nhận (Y/N) cho các thao tác xóa dữ liệu.*
- *Tất cả các thông báo lỗi đều phải đưa ra các hướng dẫn khắc phục cho người dùng.*
- *Khuyến khích vertical scrolling, hạn chế tối đa horizontal scrolling.*
- *Giao diện màn hình luôn có độ phân giải mặc định 1024×768 pixels.*
- *Đối với các quy trình tuần tự qua nhiều bước, phải có progess bar kèm thông tin chi tiết.*
- *Toàn bộ drop down list phải được sắp xếp theo thứ tự A to Z và số tăng dần.*



Câu thần chú nổi tiếng khi làm bất kỳ UI mockup nào. (Nguồn từ: Hugh Macleod)