Information technology — Software product quality

Nguyễn Minh Hiển

Ngày 26 tháng 2 năm 2024

Mục lục

6 Quality model for external and internal quality

6.1 Functionality

6.1.2 Accuracy

Accuracy (tính chính xác) của phần mềm là khả năng của phần mềm cung cấp các kết quả hoặc ảnh hưởng đúng hay chấp nhận được ở một mức độ chính xác cần thiết.

Để đo tính chính xác của phần mềm, em sẽ chia phần mềm ra các chức năng cơ bản. Với từng chức năng, đưa ra các bộ test mà đã biết trước kết quả (có thể sinh ra bằng tay hoặc bằng các công cụ khác như Excel, ...) và sau đó, chạy phần mềm và đếm số test mà phần mềm có thể chạy đúng.

Ví dụ: Về tính chính xác của một phần mềm diệt virus (AV).
Em sẽ chia chức năng chính của phần mềm này thành 2 việc: không nhận diện được mã độc và nhận diện nhầm một phần mềm hợp lệ. Về mã độc, chia làm 2 loại: mã độc mới, chưa có trong cơ sở dữ liệu của AV (để kiểm tra khả năng phân tích hành vi của AV) và mã độc phổ biến.
Các loại mã độc, phần mềm sẽ được thu thập từ các cơ sở dữ liệu đáng tin cậy trên Internet để xem AV có thể nhận diện được bao nhiêu mã độc cũng như nhận diện nhầm bao nhiều phần mềm hợp lệ.

6.1.3 Interoperability

Interoperability (tính tương tác) của phần mềm là khả năng của phần mềm tương tác với một hay nhiều hệ thống cụ thể khác nhau.
Để đo tính tương tác của phần mềm, em sẽ xem xét hiệu suất và chất lượng của phần mềm trong các tình huống tương tác khác nhau. Cụ thể hơn, em đặt các chức năng khác nhau của phần mềm vào các tình huống tương tác khác nhau nhiều lần để thống kê được xác suất thành công, thất bại, chất lượng, hiệu suất khi tương tác với các hệ thống đó.

Ví dụ: Về tính tương tác của một phần mềm mạng xã hội (SN).

Em cần thống kê được xác suất thành công, thất bại, chất lượng, hiệu suất sử dụng SN trên các hệ điều hành khác nhau (Windows, Linux, MacOS, Android, iOS, ...), các trình duyệt khác nhau (Chrome, Firefox, Safari, Opera, ...), các thiết bị khác nhau (máy tính, điện thoại, máy tính bảng, ...).
Tiếp theo đó, em cũng cần xem xét về khả năng trao đổi dữ liệu của SN đó với hệ thống bảo mật, hệ thống quảng cáo, hệ thống thanh toán, ... khác.

6.2 Reliability

6.2.1 Maturity

Maturity (tính trưởng thành) là khả năng tránh lỗi của phần mềm sau những lần gặp lỗi của phần mềm.
Điều này có thể được đo bằng số lần xảy ra lỗi và số lần lỗi được khắc phục. Cụ thể hơn ở ví dụ bên dưới.

Ví dụ: Về tính "trưởng thành" của phần mềm diệt virus Kaspersky.
Phần mềm đã xuất hiện trên thị trường từ năm 2006 với số lượng bản vá lỗi vào khoảng 100 bản được công bố trên trang web chính thức của Kaspersky. Thêm vào đó, cơ sở dữ liệu mã độc luôn được cập nhật liên tục nhằm giảm thiểu khả năng lỗi của phần mềm.

6.2.2 Fault tolerance

Fault tolerance (tính chịu lỗi) của phần mềm là khả năng của phần mềm duy trì hiệu suất ở một mức độ cụ thể trong trường hớp phần mềm gặp lỗi hoặc vi phạm giao diện cụ thể của nó.
Em đề xuất việc đo khoảng thời gian giữa 2 lần phần mềm gặp lỗi, và khoảng thời gian để phần mềm khắc phục lỗi. Cụ thể hơn: Em sẽ chọn một khoảng thời gian ngẫu nhiên rồi đo tổng thời gian lỗi, số lần lỗi. Sau đó, sử dụng các công cụ Xác suất thống kê, tính toán khoảng thời gian trung bình giữa 2 lần lỗi và khoảng thời gian trung bình để khắc phục lỗi của phần mềm đó.
khoảng thời gian trung bình giữa 2 lần lỗi càng lớn và khoảng thời gian trung bình để khắc phục lỗi càng nhỏ chứng tỏ tính chịu lỗi của phần mềm càng cao.

6.3 Usability

6.3.1 Understandability

Understandability (tính dễ hiểu) là khả năng của phần mềm cho phép người dùng hiểu được liệu phần mềm có phù hợp và có thể sử dụng trong nhiệm vụ và điều kiện cụ thể.
Một phần mềm có dễ hiểu hay không sẽ phụ thuộc vào độ phức tạp của phần mềm. Cụ thể, độ phức tạp của phần mềm có thể sử dụng các chỉ số: LOC (Lines of Code - số lượng dòng code trong mã nguồn của phần mềm), FP (function point - số lượng chức năng mà phần mềm cung cấp), ... để đo. Các chỉ số này càng cao thì phần mềm càng phức tạp và khó hiểu.

6.3.2 Learnability

Learnability (tính dễ học) là khả năng của phần mềm cho phép người dùng học cách ứng dụng phần mềm.
Để đo tính dễ học của phần mềm, em sẽ đo thời gian mà người dùng cần để học các chức năng từ cơ bản đến nâng cao của phần mềm.

Ví dụ: Về tính dễ học của một phần mềm soạn thảo (ví dụ Word).
Em sẽ thực hiện khảo sát trên 3 nhóm người dùng chưa biết gì về Word: trẻ học tiểu học, thanh niên, và trung niên, phân theo giới tính nam, nữ, và phân theo hiểu biết về máy tính. Em sẽ đo thời gian mà từng nhóm người dùng cần để học cách sử dụng các chức năng cơ bản và nâng cao của Word. Từ cách đo này có thể đánh giá được tính dễ học của Word đối với nhóm chức năng cơ bản hay nâng cao của Word, đối với từng nhóm người dùng.

6.4 Efficiency

6.4.1 Time behavior

Time behavior là khả năng của phần mềm cung cấp phản hồi, thời gian xử lý và tốc độ thông lượng phù hợp khi thực hiện các chức năng trong điều kiện đã nêu.
Rõ ràng, cách đo time behavior sẽ phụ thuộc vào chức năng cụ thể của phần mềm

và cũng được nêu rõ trong định nghĩa. Em cần đo thời gian phản hồi, thời gian xử lý và tốc độ thông lượng của phần mềm trong các khoảng thời gian ngẫu nhiên và sau đó, sử dụng các công cụ Xác suất thống kê để đánh giá time behavior của phần mềm.

6.4.2 Resource utilization

Resource utilization (tính tận dụng tài nguyên) là khả năng của phần mềm sử dụng lượng và loại tài nguyên phù hợp khi thực hiện các chức năng trong điều kiện đã nêu.
Để đo tính tận dụng tài nguyên của phần mềm, em cần đo lượng tài nguyên cần thiết để thực hiện các chức năng cụ thể của phần mềm (như lượng RAM, CPU, GPU, ổ cứng, ...) cũng như xem phần mềm đó có tận dụng đúng tài nguyên (RAM hay VRAM, CPU hay GPU, ...) không. Những tài nguyên nào được sử dụng, cùng lượng tài nguyên mà một phần mềm (hay service) đang sử dụng có thể đo đơn giản nhất trên Windows là sử dụng Task Manager để đo.

6.5 Maintainability

6.5.2 Changeability

Changeability (tính thay đổi) là khả năng của phần mềm cho phép một số chỉnh sửa cụ thể được triển khai.
Đầu tiên em cần đo về độ phức tạp của mã nguồn của phần mềm. Ở đây, em sử dụng hai chỉ số: LOC (Lines of Code - số lượng dòng code trong mã nguồn của phần mềm), FP (function point - số lượng chức năng mà phần mềm cung cấp), ... để đo.
Thứ hai, em cần xem xét về tính module hóa của phần mềm. Cụ thể là phần mềm được chia thành nhiều thành phần con riêng biệt, mỗi thành phần con có thể thay đổi mà không ảnh hưởng đến các thành phần khác. Như vậy thay đổi một chức năng chỉ ảnh hưởng đến một số module cụ thể, không ảnh hưởng đến toàn bộ phần mềm.
Thứ ba, em sẽ đo số thư viện, Framework, API, ... bên ngoài mà phần mềm sử

dụng. Các thư viện này càng được dùng nhiều thì phần mềm càng khó thay đổi (bởi không thể chỉnh sửa trực tiếp các thư viện này).
Thứ tư, cũng như quan trọng nhất là thiết kế của phần mềm. Thiết kế phần mềm càng tốt, các tài liệu đi kèm càng nhiều thì phần mềm càng dễ thay đổi một cách chính xác.

6.6 Portability

6.6.2 Installability

Installability (khả năng cài đặt) là khả năng của một phần mềm cài đặt được trên một môi trường cụ thể.
Thứ nhất, em cần phải đo số lượng yêu cầu của phần mềm này đối với hệ thống (càng nhiều yêu cầu thì càng khó cài đặt). Ví dụ như một số phần mềm yêu cầu .NET Framework, Java Runtime Environment, ... hay thậm chí là yêu cầu về cấu hình của máy tính.
Thứ hai, em cần phải đo thời gian cài đặt phần mềm và số bước cần thực hiện trên các hệ thống khác nhau. Phần mềm cài đặt nhanh và ít bước cài đặt thì Installability càng cao.
Thứ ba, em cần phải đo số lượng lỗi xảy ra trong quá trình cài đặt phần mềm và các lỗi đó được khắc phục như thế nào. Phần mềm cài đặt càng ít lỗi và lỗi được khắc phục nhanh chóng thì Installability càng cao.
Cuối cùng là những thiết lập cần thiết sau khi cài đặt phần mềm. Phần mềm càng ít thiết lập cần thiết sau khi cài đặt thì Installability càng cao.