Phân tích độ đo hướng đối tượng trong chương trình C++

Hà Thị Minh Phương

Khoa Công nghệ Thông tin và Truyền thông, Đại học Đà Nẵng
htmphuong@sict.udn.vn

Tóm tắt. Độ đo đóng vai trò rất quan trọng đối với phát triển phần mềm chất lượng tốt. Trong phát triển phần mềm ngày nay, ngôn ngữ hướng đối tượng (OO) được sử dụng dựa trên các thông tin cơ bản của chúng các tính năng như lớp, đối tượng, ẩn thông tin, kế thừa, đóng gói, trừu tượng và đa hình. Số lượng độ đo hướng đối tượng có sẵn để được sử dụng để đo lường chất lượng của các hệ thống hướng đối tượng. Một trong những độ đo phổ biến nhất trong OO là độ đo Chidamber và Kemerer (CK). Nghiên cứu này tập trung vào việc áp dụng các độ đo CK vào một các hệ thống nguồn mở được viết bằng C ++. Kết quả cho thấy chỉ có hai trong số sáu độ đo CK thì chỉ có NOC và RFC là có ảnh hưởng đến việc xảy ra lỗi trong hệ thống.

Keywords: Độ đo hướng đối tượng, lỗi phần mềm, độ đo CK, C++, mã nguồn mở.

1 Giới thiệu

Việc đo lường phần mềm đóng vai trò quan trọng trong việc giám sát chất lượng ở tất cả các giai đoạn sản xuất phần mềm. Trong công nghệ phần mềm, sư đo lường cung cấp mô tả định lượng về mức độ, số lượng, kích thước, dung lượng hoặc kích thước của một số thuộc tính của sản phẩm hoặc quy trình. Đo lường hoạt động như xác định một thước đo [1]. Chất lượng của phần mềm có thể được đo lường về các thuộc tính khác nhau như khả năng xảy ra lỗi, bảo trì, kiểm thử. Độ đo phần mềm được giới thiệu bởi Halstead năm 1972 [2]. Độ đo phần mềm cho phép sử dụng phương pháp tiếp cận kỹ thuật thực sự để phát triển phần mềm, cung cấp cơ sở định lượng và khách quan mà công nghệ phần mềm vẫn còn thiếu trong thực tế, việc sử dụng chúng ngày càng trở nên phổ biến hơn.

Độ đo phần mềm gồm độ đo cấu trúc và độ đo hướng đối tượng, độ đo hướng đối tượng mô tả các đặc điểm của sản phẩm như kích thước, độ phức tạp, tính năng, thiết kế, hiệu suất và mức độ chất lượng. Độ đo hướng đối tượng được sử dụng để nâng cao chất lượng và bảo trì phần mềm. Chúng cũng được sử dụng phổ biến hiện nay trong việc đo lường khả năng xảy ra lỗi trong các chương trình hướng đối tượng.

Có nhiều độ đo hướng đối tượng khác nhau có sẵn được nghiên cứu trong trong tài liệu [3-9] để dự đoán các thuộc tính chất lượng phần mềm. Các độ có ích đối với chương trình hướng cấu trúc thì cũng có thể là không có nhiều ảnh hưởng đối với các chương trình hướng đối tượng. Chẳng hạn như độ đo LOC được sử dụng rất nhiều trong việc phát triển các chương trình hướng cấu trúc nhưng trong chương trình hướng đối tượng thì không được dùng nhiều. Trong bài bài này chúng tôi tập trung đề cập đến sử dụng độ đo hướng đối tượng trong việc đo lường và dự đoán lỗi phần mềm. Bài báo được tổ chức thành các chương 1 giới thiệu về độ đo phần mềm, chương 2 trình bày về các độ đo hướng đối tượng, chương 3 xây dựng công cụ đo lường các độ đo hướng đối tượng, chương 4 tổng kết về các độ đo đã nghiên cứu .

2 Độ đo phần mềm

Độ đo phần mềm đóng một vai trò quan trọng trong việc thúc đẩy một yếu tố đo lường chất lượng phần mềm hiệu quả. Trong bài báo này, chúng tôi tập trung vào khai tác các độ đo phần mềm.

2.1 Độ đo Halstead and McCabe

Kích thước (Size): Độ đo đầu tiên là độ đo về kích thước được giới thiệu bởi Akiyama [10] Sau đó, nhiều nghiên cứu đo lường chất lượng phần mềm đã áp dụng số liệu này để xây dựng các dự đoán [11]. Tuy nhiên, độ đo kích thước này quá đơn giản để đo lường sự phức tạp của các sản phẩm phần mềm.

Halstead and McCabe: Các độ đo này được giới thiệu bởi McCabe năm 1976 và Halstead năm 1977. Các thuộc tính Halstead được chọn dựa trên việc đọc độ phức tạp của mã nguồn. Chúng được xác định như sau:

Ký hiệu	Mô tả
μ_1	Số lượng các toán tử riêng biệt
μ_2	Số lượng các toán hạng phân biệt
N_1	Tổng số các toán tử
N_2	Tổng số các toán hạng
μ^*_1	Số lượng tối thiểu các toán tử có thể có
μ^*_2	Số lượng tối thiểu các toán hạng có thể có

Bảng 2.1 Các mô tả cơ bản của độ đo Halstead

Độ đo Halstead được xác định bằng cách sử dụng các độ đo trên bao gồm:

Tên	Mô tả
Độ dài N= N1+N2	Độ dài chương trình (program length)
Từ vựng n = $\mu_1 + \mu_2$	Tổng số từ vựng (Vocabulary)
Độ lớn V= N* $log_2 n$	Độ lớn chương trình (program volume)
Độ phức tạp D=(μ_1 / 2) * (N_2 / μ_2)	Độ phức tap của chương trình (Difficulty)
Công sức E = V*D	Công sức viết chương trình (Effort)
Thời gian thực thi T=E/18	Thời gian thực thi yêu cầu của chương trình (Time)

Các thuộc tính của McCabe là các độ đo chu trình thể hiện độ phức tạp của một sản phẩm phần mềm. Các thuộc tính được đề xuất dựa trên giả định rằng "độ phức tạp của các con đường giữa các ký hiệu mô-đun trở nên có ý nghĩa hơn chỉ là một số lượng các ký hiệu" [12]. Khác với các thuộc tính Halstead, các thuộc tính của McCabe đo độ phức tạp của cấu trúc mã nguồn. Sau đây là ba thuộc tính phức tạp được giới thiệu bởi McCabe năm 1976.

- Độ phức tạp chu trình (Cyclomatic complexity) biểu thị bằng ν(*G*), biểu diễn cho số lượng đường dẫn độc lập tuyến tính thông qua biểu đồ luồng. ν(*G*)= *e-n+2* trong đó *G* là biểu đồ luồng, *e* đại diện cho số vòng cung, và *n* là số lượng các nút[11].
- Độ phức tạp cần thiết (Essential complexity) biểu thị bởi *e*ν(*G*), đo lường mức độ mà biểu đồ luồng có thể giảm bằng cách hủy tất cả các biểu đồ con dòng phù hợp với một lần vào – một lần ra, hoặc các cấu trúc nguyên tố D theo định nghĩa của Bieman năm 1997 *e*ν(*G*) = ν(*G*) - *m* trong đó *m* là số biểu đồ luồng con của *G*

- Thiết kế phức tạp (Design complexity) biểu thị bởi *iv*(*G)* biểu diễn cho độ phức tạp chu trình của biểu đồ luồng giảm của lớp hoặc mô-đun. Lý do giảm biểu đồ luồng là loại bỏ những phức tạp không ảnh hưởng đến mối tương quan giữa các lớp thiết kế hoặc mô-đun [13].

2.2 Độ đo hướng đối tượng

Như được liệt kê trong Bảng 2.2, một số độ đo này khá đơn giản để tính toán bằng cách đếm số thuộc tính và phương thức công cộng và riêng tư. Trong thực tế, các độ đo này được thiết kế dựa trên đặc điểm của các mô hình hướng đối tượng bao gồm thừa kế, có thể sử dụng lại, gắn kết, đóng gói và ghép nối. Do đó, tập hợp các số liệu này, còn được gọi là độ đo hướng đối tượng (OO) phù hợp để đánh giá các hệ thống hướng đối tượng.

Tên	Mô tả
NOA	Số lượng các thuộc tính
NOPA	Số lượng các thuộc tính quyền truy xuất công khai
NOPRA	Số lượng các thuộc tính quyền truy xuất riêng tư
NOAI	Số lượng các thuộc tính quyền truy xuất là thừa kế
LOC	Số dòng trong 1 lớp
NOM	Số lượng các phương thức
NOPM	Số lượng các phương thức quyền truy xuất công khai
NOPRM	Số lượng các phương thức quyền truy xuất riêng tư
NOMI	Số lượng các phương thức quyền truy xuất thừa kế

Bảng 2.2 Ví dụ về độ đo hướng đối tượng mức lớp

Ngoài các độ đo hướng đối tượng ở trên, một số độ đo khác đã được chứng minh theo các kết quả thực nghiệm là có hiệu quả để dự đoán các lỗi trong các chương trình hướng đối tượng. Đặc biệt, vào năm 1994, Chidamber và Kemerer đã sử dụng bản thể luận của Bunge làm cơ sở lý thuyết cho giới thiệu một tập hợp các độ đo CK được gọi là. Sau đó, các số liệu đã được sử dụng trong nhiều nghiên cứu để tạo ra các yếu tố dự đoán lỗi phần mềm [14].

Tên	Mô tả
WMC	Các phương thức phức tạp trong 1 lớp
DIT	Độ dài của cây thừa kế
NOC	Số lượng các lớp con
CBO	Gắn kết giữa các đối tượng
RFC	Phản hồi cho 1 lớp
LCOM	Thiếu gắn kết giữa các đối tượng

Bảng 2.3 Độ đo CK (Chidamber & Kemerer, 1994)

Các độ đo trong bảng 2.3 được mô tả như sau:

- WMC đo độ phức tạp của một lớp. Độ phức tạp của một lớp có thể được tính toán bởi các phức tạp chu trình của các phương thức của nó. Giá trị cao của WMC cho thấy lớp phức tạp cao hơn so với giá trị thấp. Vì vậy, lớp có giá trị WMC thấp hơn tốt hơn.
- Độ đo DIT là chiều dài của đường dẫn tối đa từ nút đến gốc của cây. Vì vậy, độ đo này tính toán mức độ thấp hơn một lớp được khai báo trong hệ thống phân cấp thừa kế. Độ đo này cũng đo lường số lượng lớp tổ tiên có thể ảnh hưởng đến lớp này.

- NOC độ đo này đo lường số lượng các lớp con sẽ kế thừa các phương thức của lớp cha. Kích thước của NOC xấp xỉ cho biết mức độ tái sử dụng trong một ứng dụng. Khi NOC tăng lên, số lượng trường hợp kiểm tra cũng sẽ tăng lên. Vì vậy, NOC thể hiện cho nỗ lực cần thiết để kiểm tra lớp và tái sử dụng.
- CBO Ý tưởng về độ đo này là một đối tượng được kết hợp với một đối tượng khác nếu hai đối tượng hành động với nhau. Một lớp được kết hợp với một lớp khác nếu các phương thức của một lớp sử dụng các phương thức hoặc các thuộc tính của lớp khác. Sự gia tăng CBO cho thấy khả năng sử dụng lại của một lớp sẽ giảm. Vì vậy, các giá trị CBO cho mỗi lớp nên được giữ càng thấp càng tốt.
- **RFC là số phương thức có thể được gọi để đáp ứng với một thông báo trong một lớp. Nếu RFC tăng, độ phức tạp của thiết kế tổng thể của lớp học tăng lên và trở nên khó hiểu. Mặt khác, giá trị thấp hơn cho biết tính đa hình lớn hơn.**
- LCOM độ đo này sử dụng khái niệm về mức độ giống nhau của các phương pháp. LCOM đo lường mức độ liên kết hiện tại, hệ thống được thiết kế tốt như thế nào và mức độ phức tạp của một lớp.

3 Xây dựng công cụ đo lường

3.1 Câu hỏi nghiên cứu

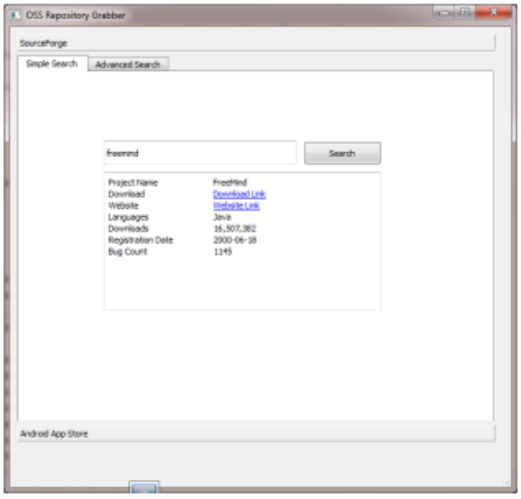
Trong bài báo này, chúng tôi muốn phân tích vai trò của độ đo mã nguồn trong việc đo lường chất lượng phần mềm. Câu hỏi nghiên cứu được chúng tôi phân tích là "những độ đo hướng đối tượng nào có ý nghĩa quan trọng và liên quan đến việc khả năng xảy lỗi của phần mềm?"

3.2 Xây dựng công cụ đo lường khả năng ra lỗi

Nghiên cứu này chủ yếu tập trung vào áp dụng bộ độ đo CK vào phần mềm OSS được viết bằng C ++. Với mục đích này, 30 chương trình C ++ đã được tải xuống từ SourceForge.net, đây là kho lưu trữ OSS lớn nhất. Để tải xuống các hệ thống từ SourceForge, một công cụ, OSSGrab, được phát triển để tự động hóa tìm kiếm và truy xuất các hệ thống. Các biến độc lập được chọn cho nghiên cứu này là tập hợp các số liệu được gọi là Chidamber và Kemerer, độ đo hướng đối tượng (độ đo CK OO).Để có được dữ liệu các độ đo, mã nguồn của hệ thống đã được tải xuống từ SourceForge được phân tích bằng công cụ trích xuất số liệu được gọi là Understand C ++. Sau khi tất cả các độ đo đã thu được, chúng được phân tích bằng sử dụng phần mềm thống kê SPSS.

3.3 Thu thập dữ liệu

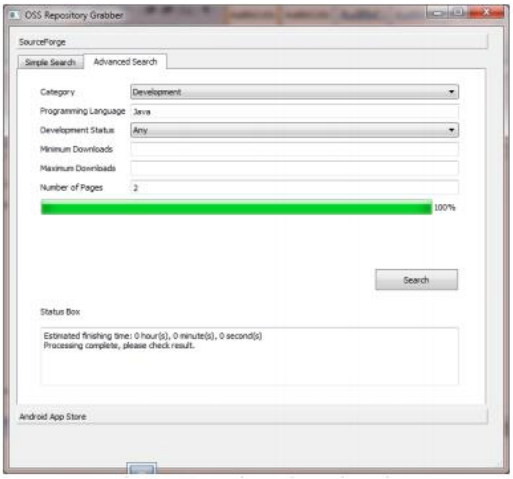
Để giảm bớt quá trình thu thập dữ liệu từ SourceForge.net, một công cụ được phát triển bằng Python và Kỹ thuật biểu thức chính quy, được gọi là OSSGrab [15]. Ứng dụng nhận được một truy vấn từ người dùng để xác định các tiêu chí để tìm kiếm trong kho lưu trữ. Truy vấn này sau đó được chuyển đến công cụ trình thu thập dữ liệu web, bắt đầu thu thập dữ liệu các trang từ API của kho lưu trữ trực tuyến tương ứng. Sau khi tải các trang công cụ trình thu thập dữ liệu web đưa nó xuống công cụ phân tích cú pháp, sau đó truy xuất dữ liệu được truy vấn từ khối lượng văn bản. Khi việc phân tích cú pháp được thực hiện xong, chương trình sẽ ghi dữ liệu được thu thập ở định dạng HTML và CSV để sử dụng cho việc nghiên cứu. Định dạng CSV cho phép người dùng thao tác với dữ liệu bằng cách sử dụng các chức năng phong phú của bảng tính.



Hình 1. Kho OSS tìm kiếm đơn giản

Đầu tiên, người dùng có thể chọn tìm kiếm các hệ thống trong SourceForge. Quá trình tìm kiếm được thể hiện trong hình 1. Hình 1 thể hiện một tìm kiếm đơn giản, trong đó người dùng cần phải xác định tên của hệ thống mà họ muốn tìm kiếm. Trình phân tích cú pháp sẽ tìm kiếm qua kho lưu trữ SourceForge và sẽ trả về kết quả cho người dùng.

Hình 2 hiển thị tùy chọn tìm kiếm nâng cao nơi người dùng có thể chọn hệ thống dựa trên danh mục, ngôn ngữ lập trình, tình trạng phát triển và số lượng tải xuống.



Hình 2. Kho tìm kiếm nâng cao

Các kết quả đầu ra được tạo ra trong cả CSV và HTML định dạng. Người dùng có thể sắp xếp đầu ra dựa trên tiêu đề của cột. Liên kết tải xuống cột sẽ kết nối người dùng tải xuống hệ thống trong SourceForge. Điều này sẽ cung cấp quyền truy cập nhanh vào hệ thống và người dùng có thể trực tiếp tải xuống hệ thống. Từ quan điểm nghiên cứu, điều này cơ sở sẽ cung cấp cho các nhà nghiên cứu nhiều lựa chọn về hệ thống để lựa chọn. Trong kỹ thuật phần mềm thực nghiệm, các nhà nghiên cứu cần tìm nhiều dữ liệu nhất có thể, đặc biệt khi họ muốn xây dựng mô hình dự báo, để đảm bảo rằng các mô hình có thể được tổng quát hơn cho dân số nói chung. Sau đó, mã nguồn của các hệ thống được chọn sẽ là được tải xuống và thực thi thông qua một công cụ có tên là Understand [16], sẽ tạo ra một tập hợp các số liệu i. e., CBO, WMC,

DIT, LCOM, NOC và RFC. Kết quả sau được phân tích sử dụng một gói thống kê, SPSS.

Thống kê mô tả cho các biến độc lập là thể hiện trong Bảng 1. Số lượng quan sát hoặc cỡ mẫu cho nghiên cứu này là 30 hệ thống. Cột "Std. Dev", "Min", "Max" đại diện cho giá trị trung bình, độ lệch chuẩn, giá trị tối thiểu và tối đa cho mỗi độ đo được xem xét, tương ứng.

Bảng 1. Thống kê kết quả theo SPSS

Độ đo	Mean	Std.Dev.	Min	Max
CBO	0.63	48.33	5.90	8.28
WMC	0.24	21	12.47	4.71
LCOM	14.50	367.67	54.83	60.25
DIT	0.18	24	1.55	4.26
NOC	0.00	27.33	1.31	4.92
RFC	3.33	88.65	23.48	15.67

Phân tích ban đầu về dữ liệu cho thấy phân phối là không bình thường. Do đó, để phân tích tương quan, Spearman phân tích tương quan đã được thực hiện. Kết quả tổng thể của phân tích spearman được thể hiện trong Bảng 1.

Trong Bảng 2, hàng trên đại diện cho các giá trị cho hệ số tương quan Spearman giữa hai biến, trong khi hàng dưới cùng (trong ngoặc đơn) đại diện cho giá trị p cho tương quan. Kết quả trong Bảng II cho thấy chỉ có RFC và NOC có ý nghĩa quan trọng trong việc dự đoán các khuyết tật (Spearman corr. = 0.495, p-value = 0.005) và (Spearman corr. = 0.36, p-value = 0.05).

Bảng 2. Phân tích tương quan Spearman giữa các biến

Độ đo	Lỗi	CBO	RFC	NOC	WMC	LCOM
Lỗi	1	0.038	0.495	0.36	0.17	0.058
CBO	-	(0.84)	(0.005)	(0.05)	(0.369)	(0.76)
		1	0.528	0.306	0.278	0.423
RFC	-	-	(0.003)	(0.10)	(0.137)	(0.02)
				0.38	0.533	0.244
NOC	-	-	-	1	-0.305	-0.032
					(0.101)	(0.867)
WMC	-	-	-	-	1	0.544
						(0.002)
LCOM	-	-	-	-	-	1

4 Kết luận

Việc áp dụng các độ đo thiết kế hướng đối tượng (OO) để đo lường chất lượng của các hệ thống nguồn mở đã giúp hiểu các mối quan hệ của các số liệu với khả năng xảy ra lỗi của các hệ thống. Bài viết này tìm cách điều tra các số liệu có thể được sử dụng để dự đoán lỗi trong các hệ thống OO, đặc biệt là những cái được viết bằng C ++. Kết quả cho thấy chỉ RFC và NOC có ý nghĩa quan trọng trong việc dự đoán khả năng xảy ra lỗi. Nghiên cứu này có thể được mở rộng hơn nữa để so sánh giá trị số liệu cho các hệ thống được viết bằng C ++ và hệ thống Java.

Tài liệu tham khảo

1. IEEE Standard 1990, IEEE Standard Glossary of Software Engineering Terminology, No. STD 610-12-1990, 1990.
2. Fabrizio Riguzzi: A Survey of Software Metrics, July 1996, DEIS Technical Report no. DEIS-LIA-96-010 (1996).
3. S. Chidamber and C. Kemerer: "A Metrics Suite for Object-Oriented Design," IEEE Trans. Soft Ware Eng., Vol.20, No.6, pp.476-493 (1994).
4. L.Briand, P. Devanbu, W. Melo: "An investigation into coupling Measures for C++," In Proceedings of the 19th International Conference on Software Engineering.
5. J. Bansiya and C. Davis: "A Hierarchical Model for Object-Oriented Design Quality Assessment," IEEE Trans. Software Eng., Vol.28, No.1, pp.4-17 (2002).
6. F. Brito e Abreu and W. Melo: "Evaluating the Impact of Object-Oriented Design on Software Quali ty," Proceedings Third Int'l Software Metrics Symposium, pp.90-99 (1996).
7. M.Lorenz and J. Kidd: "Object-Oriented Software Metrics," Prentice-Hall (1994).
8. W. Li and W. Henry: "Object-Poiented Metrics that Predict Maintainability," In Journal of Software and Sytems, Vol.23, pp.111-122 (1993).
9. M.Cartwright and M. Shepperd: "An empirical investigation of an object-oriented software system," IEEE Transactions on Software Engineering, Vol.26, No.8,1999, pp.786-796
10. Akiyama, F.: An Example of Software System Debugging. In Proceedings ofthe International Federation of Information Processing Societies Congress, 1971, pages 353–359.
11. D'Ambros, M., Lanza, M., & Robbes, R.: Evaluating defect prediction approaches: a benchmark and an extensive comparison. Empirical Software Engineering, 17(4-5), 531-577 (2012)
12. Menzies, T., DiStefano, J., Orrego, A., & Chapman, R.: Assessing predictors of software defects. Paper presented at the Proc. Workshop Predictive Software Models (2004).
13. McCabe, T. J.: A complexity measure. Software Engineering, IEEE Transactions on (4), 308-320 (1976).
14. Bacchelli, A., D'Ambros, M., & Lanza, M.: Are popular classes more defect prone?. In Fundamental Approaches to Software Engineering, (pp. 59-73). Springer Berlin Heidelberg (2010).
15. N. S. A. A. Bakar and I. Mahmud: "OSSGrab: Software repositories and app store mining tool," Lecture Notes in Software Engineering (LNSE), vol. 5, pp. 55-61 (2013).
16. Scitools. [Online]. Available: http://www.scitools.com/index.php