ĐÁNH GIÁ SỰ KẾT HỢP CHUYÊN VIÊN KIỂM THỬ PHẦN MỀM VỚI KIỂM THỬ TỰ ĐỘNG TRONG MÔ HÌNH PHÁT TRIỂN PHẦN MỀM CỦA SCRUM

Lê Vũ [1, 3*], Đặng Đức Nam [2], Phạm Trung Đức [3], Dương Phước Đạt [3,4], Nguyễn Mậu Hân [3]

[1] Trường Đại học Sư phạm Kỹ thuật, Đại học Đà Nẵng

[2] Công ty Axon Active, Chi nhánh Đà Nẵng

[3] Trường Đại học Khoa học, Đại học Huế

[4] Khoa Du lịch, Đại học Huế

*Email: levuvn@gmail.com

Ngày nhận bài: 4/12/2017; ngày hoàn thành phản biện: 21/5/2018; ngày duyệt đăng: 8/6/2018

TÓM TẮT

Phương thức phát triển phần mềm Scrum là một mô hình làm việc hiệu quả cho các nhóm lập trình viên. Trong bài báo này, chúng tôi đánh giá sự kết hợp việc đưa Chuyên viên kiểm thử phần mềm tham gia vào làm việc ngay từ đầu với các nhóm lập trình viên và tiến hành kiểm thử tự động với hỗ trợ của **BDD (Behavior Driven Development)** khi vận hành Scrum. Trong bài báo này, chúng tôi đánh giá, so sánh tốc độ, năng suất làm việc của các nhóm áp dụng sự kết hợp kiểm thử với các nhóm thực hiện theo mô hình truyền thống của Scrum.

Từ khóa: Scrum, Chuyên viên kiểm thử phần mềm, BDD, phát triển phần mềm.

1. ĐẶT VẤN ĐỀ

Theo truyền thống, quá trình phát triển phần mềm dựa vào việc sử dụng mô hình "Thác nước". Sau đó, các phương pháp phát triển phần mềm linh hoạt (Agile Software Development – gọi tắt là Agile) đã được sử dụng để giải quyết những thách thức của việc quản lý các dự án phức tạp đang trong giai đoạn phát triển. Phương pháp luận Agile là một nhóm các phương pháp gia tăng và lặp đi lặp lại hiệu quả hơn và được sử dụng trong quản lý dự án. Scrum là phương thức quản lý dự án phát triển phần mềm theo mô hình của Agile được sử dụng phổ biến và mang lại nhiều hiệu quả trong bối cảnh hiện nay [1]. Mục tiêu của Scrum là tối ưu hóa quá trình phát triển phần mềm bằng cách xác định các nhiệm vụ, quản lý thời gian hiệu quả hơn và thiết lập các nhóm lập trình viên. Tuy nhiên theo mô hình truyền thống của Scrum thì khi quá trình phát triển phần mềm hoàn thành thì các lập trình viên (Developers) thực hiện kiểm thử. Về lâu dài thì việc làm này không có tính ổn định, các lập trình viên có thể lập

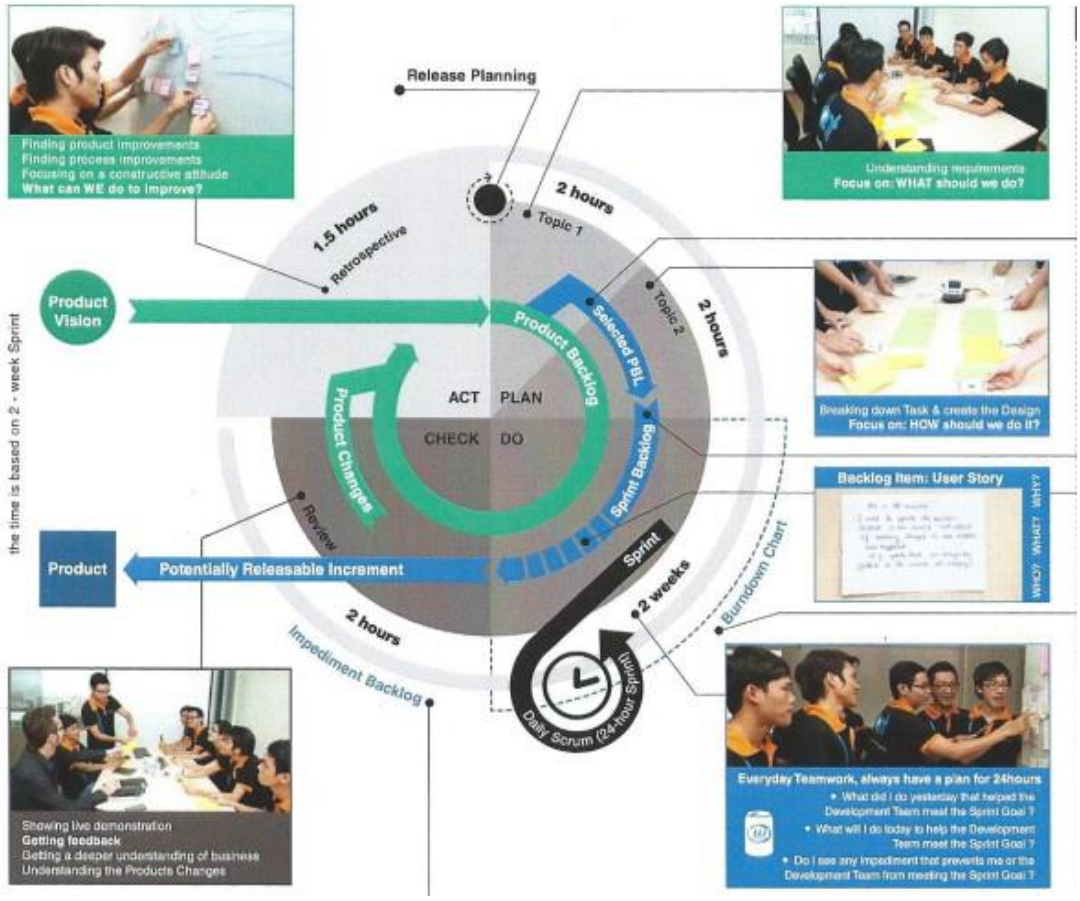
trình các chức năng của phần mềm trong một khoảng thời gian ngắn, chẳng hạn là 2 tuần, nhưng khi làm xong thì vẫn không thấy sự phù hợp, sản phẩm chưa đạt được mong muốn của khách hàng. Điều này do nhiều nguyên nhân: lập trình viên không hiểu được yêu cầu của khách hàng; người test chức năng sản phẩm với PO (Product Owner), Dev không hiểu nhau, tức là có xung đột ý tưởng trong quá trình phát triển dự án phần mềm. Mỗi chủ thể trong quá trình thực hiện phát triển phần mềm theo một hướng khác nhau, Khách hàng – PO – Dev không hiểu ý nhau; các Dev trong nhóm không hiểu nhau, từng người xây dựng các chức năng riêng, nhưng không có sự liên quan với nhau, sản phẩm của những người trong nhóm không có sự kế thừa, liên kết với nhau, gây nên sự xung đột trong sản phẩm, hoặc dự án phần mềm đang làm mà có thêm một người mới bổ sung vào nhóm thì người mới vào sẽ mất rất nhiều thời gian để hiểu nghiệp vụ.

Từ đó, trong nghiên cứu này chúng tôi áp dụng một phương pháp khi vận hành Scrum bằng cách đưa ra một famework có sự kết hợp, đó là đưa Chuyên viên kiểm thử phần mềm vào làm việc với nhóm lập trình viên ngay từ đầu, thay vì phải đợi đến hết sprint mới kiểm thử và chúng tôi áp dụng kiểm thử tự động với sự hỗ trợ của mô hình BDD.

Bài báo được được tổ chức bao gồm các phần chính như sau: phần 1 giới thiệu tính cấp thiết; phần 2 trình bày mô hình Srum và BDD; phần 3 giới thiệu sự kết hợp kiểm thử với mô hình Scrum, thảo luận, đánh giá và cuối cùng là kết luận, hướng phát triển.

2. SCRUM VÀ BDD

2.1 Mô hình Scrum



Hình 1. Mô hình Scrum

Scrum bao gồm các nhóm Scrum, sự kiện, các tạo tác và các quy tắc. Các quy tắc rất cần thiết để ràng buộc các nhóm, các sự kiện và tạo tác gắn kết với nhau trong suốt dự án. Scrum cung cấp một cấu trúc để giải quyết các vấn đề trong một dự án. Các phần sau giải thích cụ thể [2].

2.1.1. Nhóm Scrum

Nhóm Scrum bao gồm một Trưởng dự án (Product Owner – PO), một Scrum Master và các thành viên của nhóm phát triển. Các nhóm tự tổ chức và có chức năng chéo, có quyền kiểm soát dự án và biết làm thế nào để hoàn thành các mục tiêu của dự án. Tất cả các thành viên của nhóm dự án Scrum, PO, Scrum Master và các bên liên quan có rất nhiều cơ hội để kiểm tra và điều chỉnh sản phẩm trong suốt dự án và cuối cùng tạo ra sản phẩm tốt nhất. Vì các framework của mô hình Scrum cho phép nhận các phản hồi liên tục và qua đó có thể nhanh chóng điều chỉnh. Nhóm Scrum cung cấp sản phẩm lặp đi lặp lại và từng bước, tối đa hóa phản hồi mà nhóm nhận được. Sau đây là những mô tả về vai trò khác nhau của nhóm Scrum:

- PO quản lý Product Backlog (nơi lưu trữ danh sách các tính năng mong muốn của sản phẩm), danh sách yêu cầu của sản phẩm và tối đa hoá giá trị

của dự án. Vai trò của PO cũng bao gồm việc giải thích các mục Product Backlog của dự án và những mục tiêu của dự án cho các Dev, đảm bảo việc cả nhóm sẽ hiểu được những mục tiêu của dự án ở mức độ chắc chắn nhất.

- Scrum Master quản lý Backlog Sản phẩm và hướng dẫn các Dev tạo các mục trong Product Backlog rõ ràng. Scrum Master cũng đảm nhiệm vai trò liên lạc với các thành viên trong nhóm để đảm bảo cả nhóm nghiên cứu hiểu được các kế hoạch dài hạn của dự án. Ngoài ra, Scrum Master làm việc với các Scrum Masters khác để tăng hiệu quả tổ chức của Scrum.
- Nhóm phát triển (hay còn gọi là nhóm lập trình viên - Dev) chịu trách nhiệm triển khai và phân phối sản phẩm có thể hoàn tất được vào cuối mỗi "Sprint", tức là khoảng thời gian (gọi tắt là hộp thời gian) để tạo ra sự gia tăng khả năng sử dụng được của sản phẩm. Nhóm kiểm soát việc thực hiện sản phẩm cuối cùng; các thành viên của Nhóm phát triển quản lý công việc của nhóm và tự tổ chức, nguyên tắc là không được nhóm lại thành các nhóm phụ. Quy mô của nhóm là một vấn đề quan trọng; một nhóm nhỏ có thể gặp các vấn đề thiếu kỹ năng, trong khi một nhóm lớn có thể phải chịu sự phức tạp phát triển.

2.1.2. Các tạo tác của Scrum

Các tạo tác Scrum bao gồm Product Backlog, Sprint Backlog và định nghĩa về "hòan thành" sản phẩm là sau mỗi lần gia tăng, tổng của các mục Product Backlog được hoàn thành thông qua Sprint.

Product Backlog chứa danh sách các yêu cầu, chức năng, cải tiến và các điều chỉnh cần thiết trong sản phẩm. Danh sách này cho thấy các chức năng của sản phẩm ttheo cách nhìn kỹ thuật và kinh doanh. PO chịu trách nhiệm tạo ra danh sách và giải thích quan điểm của dự án cho nhóm. Sprint Backlog là danh sách các mục trong Product Backlog đã được chọn cho Sprint cụ thể. Nhóm phát triển mô tả các chức năng của sản phẩm sẽ được thực hiện trong Sprint tiếp theo và công việc cần thiết. Trong Sprint, nếu Nhóm phát triển nhận thấy rằng có nhiều công việc cần thiết, nhóm sẽ bổ sung thêm công việc cho Sprint Backlog. Công việc còn lại trong Sprint Backlog có thể được theo dõi bởi nhóm để quản lý tiến trình của Sprint.

2.1.3. Các sự kiện Scrum

Scrum sử dụng các sự kiện đóng hộp thời gian cùng với quá trình phát triển dự án và lập kế hoạch dự án. Các sự kiện trong Scrum được thiết kế để kiểm tra tạo tác và để thích ứng với các phương pháp mới dùng trong giải quyết các quy trình của dự án. Mục tiêu của những sự kiện là để minh bạch, thích nghi và kiểm tra trong quá trình phát triển [2]. Hình 1 cho thấy các nội dung của mỗi sự kiện Scrum.

Sprint là trung tâm của quá trình vận hành Scrum. Đây là một hộp thời gian để tạo ra một sản phẩm có thể sử dụng được. Mỗi Sprint có thể được coi là dự án một tháng với kế hoạch về những gì cần phải được xây dựng và cần được xây dựng như thế nào.

Nhóm Scrum lập kế hoạch các mục tiêu của mỗi Sprint, cùng với quá trình hoàn thiện sản phẩm, lập kế hoạch cuộc họp trong sprint. Mục tiêu tổng thể của mỗi lần sprint là tạo ra một sản phẩm có thể sử dụng được và có khả năng phát hành, được gọi là sản phẩm "hoàn tất". Các thành viên của nhóm Scrum thảo luận và có một sự hiểu biết chung về những gì cần thiết để tạo nên một sản phẩm "hoàn tất". Thời gian họp theo kế hoạch Sprint thường là 8 giờ, xảy ra mỗi tháng một lần trước mỗi Sprint [2]. Ngoài cuộc họp này, có cuộc họp Scrum hàng ngày 15 phút, trong đó các thành viên trong nhóm cập nhật về tiến triển của công việc đang làm, các mục tiêu dự định cho cuộc họp tiếp theo và những khó khăn, vướng mắc đã trải qua mỗi ngày.

Vào cuối mỗi Sprint, có một cuộc họp đánh giá Sprint được tổ chức để thảo luận về những gì từng thành viên trong nhóm làm trong quá trình lặp lại để tạo ra sản phẩm. Cuộc họp này có thể là một cuộc trình diễn demo sản phẩm cho PO, hoặc đôi khi với cả PO và khách hàng. Dựa trên đó và bất kỳ thay đổi nào đối với Product Backlog trong Sprint, những người tham dự cộng tác trên những điều có thể được thực hiện để tối ưu hóa giá trị. Đây là một cuộc họp không chính thức, nhằm mục đích gợi ra phản hồi và thúc đẩy sự cộng tác.

Sau cuộc họp xem lại Sprint và trước Sprint tiếp theo, tổ chức cuộc họp nhìn lại quá khứ của Sprint để kiểm tra về truyền thông, nguồn nhân lực, quá trình xử lý, công cụ và xác định những cải tiến tiềm năng cho Sprint trong tương lai. Cuộc họp này thường mất vài giờ. Đối với Sprint ngắn hơn, sự kiện thường ngắn hơn. [2].

2.2 Mô hình BDD

BDD (Behavior Driven Development) là một quá trình phát triển phần mềm dựa trên phương pháp Agile. BDD là sự mở rộng của TDD (Test Driven Development). Thay vì tập trung vào phát triển phần mềm theo hướng kiểm thử, BDD tập trung vào phát triển phần mềm theo hướng hành vi [3] [4].

Dựa vào yêu cầu các kịch bản test sẽ được viết trước dưới dạng ngôn ngữ tự nhiên và dễ hiểu nhất sau đó mới thực hiện cài đặt mã nguồn. Thay vì chờ đợi sản phẩm hoàn thành và kiểm thử, Chuyên viên kiểm thử phần mềm tham gia vào quá trình xây dựng mã nguồn với vai trò phân tích và xây dựng hệ thống kịch bản kiểm thử dưới góc độ ngôn ngữ tự nhiên dễ hiểu từ các yêu cầu.

Đồng thời, Chuyên viên kiểm thử phần mềm giúp đỡ lập trình viên trong việc giải thích và đưa ra các phương án xây dựng mã nguồn mang tính thực tiễn với người dùng ngay trước khi bắt tay xây dựng. Lập trình viên liên hệ mật thiết với Chuyên

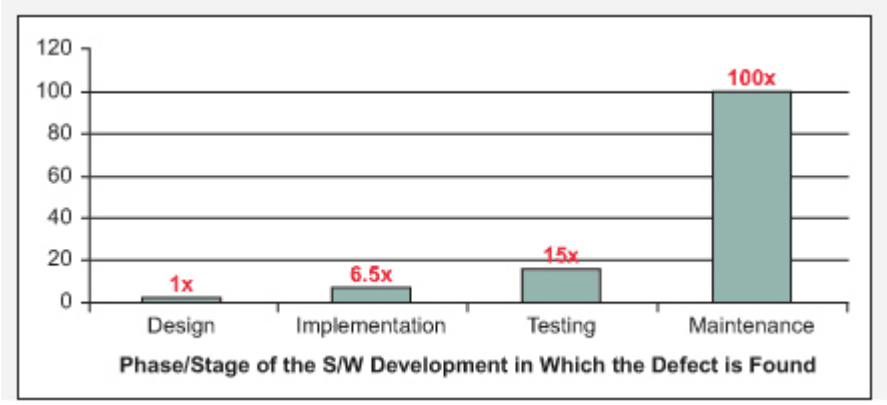
viên kiểm thử phần mềm và xây dựng mã nguồn với những phương án mà Chuyên viên kiểm thử phần mềm cung cấp theo mô hình TDD.

BDD giúp xác định đúng yêu cầu của khách hàng: tài liệu được viết dưới dạng ngôn ngữ tự nhiên, bất kỳ đối tượng nào cũng có thể hiểu được. Khi đọc tài liệu này, khách hàng có thể dễ dàng nhận biết được lập trình viên có hiểu đúng yêu cầu của họ không và có phản hồi kịp thời. Đồng thời BDD là tài liệu sống của dự án: tài liệu này luôn được cập nhật khi có bất kỳ sự thay đổi nào nên tất cả các thành viên sẽ không bị chệch thông tin khi phát triển hệ thống. Áp dụng BDD giúp nâng cao chất lượng phần mềm, tạo ra sản phẩm hữu ích: vì phát triển phần mềm theo hướng hành vi nên có thể tập trung vào việc tạo ra sản phẩm đúng với yêu cầu của khách hàng nhưng vẫn hữu ích cho người dùng.

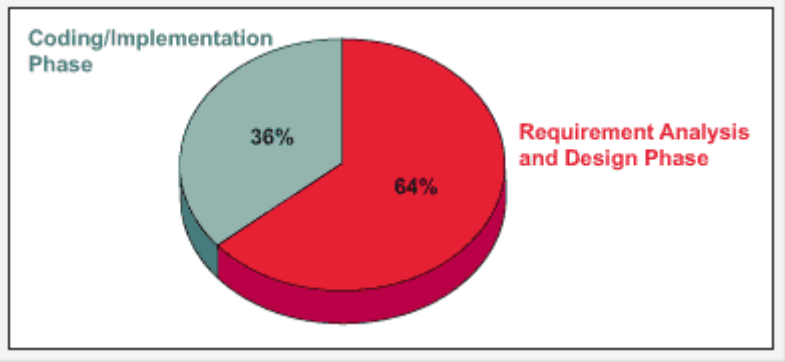
3. ĐÁNH GIÁ SỰ KẾT HỢP KIỂM THỬ TROG VẬN HÀNH MÔ HÌNH SCRUM

3.1 Chi phí sửa lỗi phần mềm

Trong báo cáo của IEEE Software [5] cho rằng đánh giá nghiêm ngặt thường loại bỏ tới 90% lỗi từ một sản phẩm phần mềm trước khi chạy thử nghiệm đầu tiên. Các phần đánh giá nghiêm ngặt có hiệu quả hơn và tiết kiệm chi phí hơn bất kỳ phương pháp loại bỏ lỗi nào khác, bao gồm cả thử nghiệm. Nhưng không thể và không nên thay thế testing. Viện Khoa học Hệ thống của IBM (IBM Systems Sciences Institute) đã thống kê và cho rằng chi phí để sửa lỗi được tìm thấy sau khi phát hành sản phẩm gấp 4 đến 5 lần so với một phát hiện trong giai đoạn thiết kế và hơn 100 lần khi được xác định trong giai đoạn bảo trì (Hình 2) [6]. Việc rà soát hoặc kiểm tra trong giai đoạn thiết kế có thể xác định được một số lượng đáng kể các lỗi. Theo Crosstalk, Journal of Defense Software Engineering, hầu hết các thất bại trong các sản phẩm phần mềm là do lỗi trong giai đoạn yêu cầu và các giai đoạn thiết kế chiếm tới 64% tổng chi phí khiếm khuyết (Hình 3).



Hình 2. *Chi phí tương đối để sửa lỗi phần mềm (Nguồn: IBM Systems Sciences Institute)*



Hình 3. *Xuất xứ của các khiếm khuyết phần mềm (Nguồn: Crosstalk, Journal of Defense Software Engineering)*

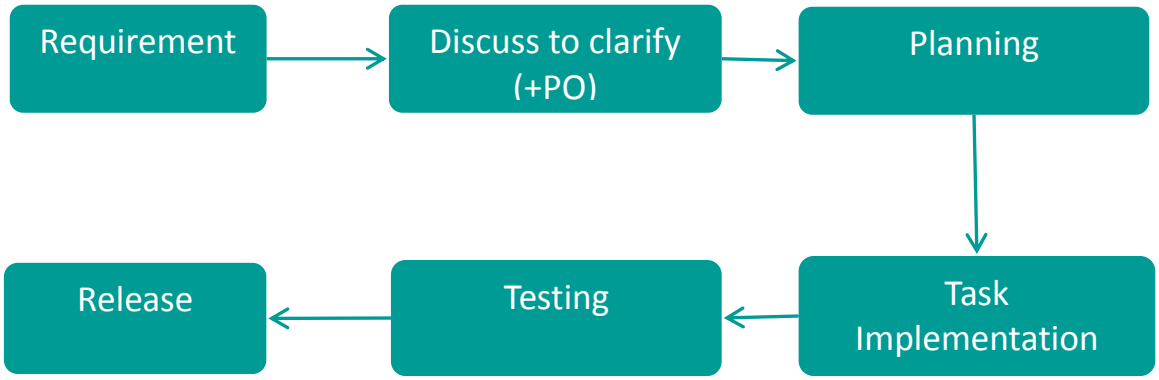
Chi phí sửa lỗi phần mềm là thấp nhất trong giai đoạn yêu cầu của vòng đời phát triển phần mềm. Điều này là do có rất ít sản phẩm ngay khi bắt đầu dự án để sửa chữa nếu có lỗi. Khi dự án chuyển sang giai đoạn tiếp theo của phát triển phần mềm, chi phí sửa lỗi tăng lên đáng kể vì có nhiều sản phẩm bị ảnh hưởng bởi sự điều chỉnh lỗi, chẳng hạn như tài liệu thiết kế hoặc mã của nó. Tỉ lệ chi phí lỗi phần mềm được so với định mức đưa ra khi khởi động dự án, hay nói cách khác là % trên tổng giá trị đầu tư của dự án, chi phí lỗi phần mềm tùy thuộc vào giá trị của từng dự án khác nhau. Xem Bảng 1 về chi phí liên quan để sửa lỗi ở các giai đoạn khác nhau trong vòng đời phát triển.

Bảng 1. Chi phí lỗi phần mềm (Nguồn: IBM Systems Sciences Institute)

Giai đoạn tìm thấy lỗi	**Tỉ lệ chi phí**
Yêu cầu	1
Thiết kế	3 – 6
Coding	10
Kiểm tra đơn vị/Tích hợp	15 – 40
Kiểm tra hệ thống/Chấp nhận	30 – 70
Thành sản phẩm	40 – 1000

3.2 Sự kết hợp và đánh giá

Từ những vấn đề đặt ra về chi phí sửa lỗi phần mềm, chúng tôi sử dụng phương pháp kết hợp chuyên viên kiểm thử phần mềm với kiểm thử tự động cùng với quá trình vận hành Scrum theo famework dưới đây.



Hình 4. Famework áp dụng BDD khi vận hành mô hình Scrum

Khi có yêu cầu do khách hàng đưa ra, sau đó tổ chức họp nhóm để làm rõ các yêu cầu của khách, thời điểm này Chuyên viên kiểm thử phần mềm cùng làm việc với nhóm phát triển và cùng nắm các yêu cầu của khách hàng; tiếp theo đó là lập kế hoạch, tại giai đoạn này xác định làm sản phẩm trong bao lâu, làm những cái gì, thời điểm làm xong, thời điểm test. Ý tưởng của việc áp dụng BDD khi vận hành Scrum là khi có yêu cầu của khách hàng và sau khi họp phiên đầu tiên của nhóm xong thì Chuyên viên kiểm thử phần mềm sẽ viết ra các kịch bản chạy rất rõ ràng của phần mềm, sử dụng các trường hợp kiểm thử và kịch bản kiểm thử để tạo ra các kịch bản kiểm thử tự động. Khi sử dụng các kịch bản kiểm thử thì cả Chuyên viên kiểm thử phần mềm và lập trình viên đều tường minh các công việc của Sprint và hoàn toàn yên tâm về sản phẩm "hoàn tất", công sức kiểm tra lại sản phẩm gần như bằng 0, đảm bảo chất lượng sản phẩm.

BDD yêu cầu Chuyên viên kiểm thử phần mềm tham gia ngay giai đoạn lấy yêu cầu sẽ giúp cho việc tìm ra bug sớm hơn, chi phí thấp hơn để sửa lỗi phần mềm. Ngoài ra BDD còn tự động hoá quá trình, loại bỏ lỗi mà do yếu tố con người trong giai đoạn acceptance test, giúp giảm thiểu rủi ro.

Khi thực nghiệm mô hình Scrum, một khái niệm mới về ước lượng được sử dụng tại các nhóm phát triển phần mềm là điểm để tính tốc độ làm việc. Thay vì tính theo ngày công thì các công việc diễn ra của từng Sprint được tính bằng điểm. Điểm này do từng nhóm phát triển phần mềm qui định, không có sự giống nhau giữa các nhóm. Ví dụ: hoàn thiện giao diện login được tính ra điểm; hay tùy theo độ phức tạp, độ khó trong thiết kế giao diện nhập liệu thì sẽ tương ứng với số điểm…

Trong thực tế, khi vận hành mô hình Scrum và mô hình kết hợp kiểm thử với Scrum vào các dự án phát triển phần mềm tại công ty Axon Active, chúng tôi thu thập được số liệu về tốc độ làm việc, số lỗi sản phẩm, số lượt khiếu nại của khách hàng như sau. [*Trong nhóm viết bài báo của chúng tôi có thành viên làm quản lý tại công ty Axon Active nên chúng tôi có những số liệu chính xác và những thông tin về qui trình sản xuất phần mềm*

theo chuẩn Scrum có sự kết hợp của kiểm thử đã được các chuyên gia kinh nghiệm nhiều năm thử nhiệm nên kết quả là hoàn toàn chính xác, trung thực, theo chuẩn của thế giới. Tiêu chí cho điểm chúng tôi đã giải thích ở trên]

Bảng 2. Số liệu năng suất làm việc của các nhóm phát triển trong năm 2016

Tên nhóm	**Vận hành mô hình Scrum**	**Vận hành mô hình kết hợp kiểm thử với Scrum**	**Tốc độ làm việc (đơn vị tính: điểm)**	**Số lỗi sản phẩm**	**Số lượt khiếu nại khách hàng**
PGDN1	X		80	30	8
PGDN2		X	92	3	0
BDNA	X		87	23	5
BKQN		X	95	4	0
NDCT1		X	94	2	0
NDCT2		X	97	2	0
THBD		X	93	3	1
HKVN	X		71	26	6
KKVN		X	98	1	0

Qua bảng trên, chúng tôi nhận thấy tốc độ làm việc của các nhóm vận hành mô hình kết hợp kiểm thử với Scrum có năng suất cao hơn, do kịch bản đã được các Chuyên viên kiểm thử phần mềm cài đặt vào hệ thống giúp cho hệ thống luôn được xây dựng lại tự động khi có sự thay đổi, từ đó tiết kiệm được công sức hơn rất nhiều so với mô hình Scrum truyền thống. Tốc độ làm việc của các nhóm vận hành mô hình Scrum chậm hơn mô hình kết hợp kiểm thử với Scrum nhiều, do phải tốn khá nhiều thời gian để giải quyết các "xung đột", không hiểu ý nhau trong quá trình thực hiện các Sprint. Theo quan sát trong thực tế tại công ty, chúng tôi cũng nhận thấy ở mô hình kết hợp kiểm thử với Scrum, thì ở giai đoạn đầu của dự án, tốc độ làm việc của Chuyên viên kiểm thử phần mềm thấp khoảng 01 lượt Sprint, sau đó thì đi vào ổn định ở các Spint kế tiếp [*thành viên của nhóm viết bài báo đã thống kê trong quá trình làm công tác quản lý tại công ty Axon Active, mô hình Scrum là mô hình được rút ra từ kinh nghiệm thực tiễn và hiện áp dụng trong sản xuất phần mềm, đã được thực tiễn chứng minh nên chúng tôi cho rằng số liệu thu thập và những phản ánh của khách hàng đã chứng minh tính hiệu quả của nó*]. Hai số liệu có tầm quan trọng trong hai mô hình này, đó là số lỗi sản phẩm và số lượt khiếu nại của khách hàng, mô hình Scrum truyền thống chiếm một tỷ trọng lớn hơn mô hình kết hợp kiểm thử với Scrum rất nhiều. Những khiếu nại của khách hàng dành cho mô hình kết hợp kiểm thử với Scrum hầu như không có.

4. KẾT LUẬN

Từ việc đánh giá mô hình phát triển phần mềm Scrum ở trên, chúng tôi nhận thấy mô hình Scrum có kết hợp Chuyên viên kiểm thử phần mềm và hỗ trợ của mô hình BDD hoạt động hiệu quả hơn mô hình Scrum truyền thống. Điều này góp phần đảm bảo tính năng của sản phẩm, sản phẩm chất lượng hơn, quan trọng là đã giải quyết được vấn đề không rõ ràng về yêu cầu của các thành phần trong dự án phần mềm. Trong thời gian đến, chúng tôi sẽ tiếp tục nghiên cứu mô hình Scrum để đưa ra những giải pháp cải tiến hiệu quả hơn nữa, ví dụ như cải tiến qui trình lập kịch bản automation, cải tiến các công cụ kiểm thử phần mềm để giúp cho việc quản lý dự án được tốt hơn, chất lượng hơn.

TÀI LIỆU THAM KHẢO

[1] M. Paasivaara and C. Lassenius, "Scaling Scrum in a Large Globally Distributed Organization: A Case Study," in *2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*, 2016, pp. 74–83.

[2] K. Schwabe and J. Sutherland, "The Scrum Guide ™," 2013. [Online]. Available: https://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf.

[3] "Behavior Driven Development." [Online]. Available: http://behaviour-driven.org.

[4] I. Lazăr, S. Motogna, and B. Pârv, "Behaviour-Driven Development of Foundational UML Components," *Electron. Notes Theor. Comput. Sci.*, vol. 264, no. 1, pp. 91–105, 2010.

[5] "The Rising Costs of Defects," 2014. [Online]. Available: https://www.seguetech.com/rising-costs-defects/.

[6] "DevOps: Shift left with continuous testing by using automation and virtualization," *IBM Systems Sciences Institute*. [Online]. Available: https://www.ibm.com/cloud/garage/experience/deliver/dibbe_edwards_devops_shift_left/.

EVALUATING THE COMBINATION OF MANUAL SOFTWARE TESTERS AND AUTOMATIC TEST WHILE APPLYING SCRUM AGILE FRAMEWORK

Le Vu [1, 3*], Dang Duc Nam [2], Pham Trung Duc [3], Duong Phuoc Dat [3,4], Nguyen Mau Han [3]

[1] University of Technology and Education, The University of DaNang

[2] Axon Active VietNam, Da Nang Branch

[3] University of Sciences, Hue University

[4] School of Hospitality and Tourism, Hue University

*Email: levuvn@gmail.com

ABSTRACT

Scrum is the highly efficient agile software development framework (or methodology) for developers. In this article, we evaluate the combination of software testers joining the project with developers from the start and implement the automatic test with the support of BDD (Behavior Driven Development) while applying Scrum methodology. We also evaluate and compare development speed and productivity of developer groups utilizing the combination test with groups following traditional Scrum model.

Keywords: BDD, Scrum, software development, tester.

Nguyễn Mậu Hân sinh năm 1957 tại Thừa Thiên Huế. Ông tốt nghiệp cử nhân ngành Toán lý thuyết năm 1981 và thạc sĩ chuyên ngành Khoa học máy tính năm 1998. Ông nhận bằng tiến sĩ tại viện Công nghệ thông tin, Hà Nội năm 2003 và được phong hàm Phó Giáo sư năm 2013. Từ năm 1994 đến nay, ông là giảng viên tại khoa Công nghệ thông tin, Trường Đại học Khoa học, Đại học Huế.

Lĩnh vực nghiên cứu: Xử lý song song và phân tán, tính toán lưới và điện toán đám mây.

Lê Vũ sinh năm 1979 tại Đà Nẵng. Nhận bằng Thạc sĩ Khoa học máy tính tại Trường Đại học Công nghệ Thông tin, Đại học Quốc gia Thành phố Hồ Chí Minh năm 2012, và đang là nghiên cứu sinh ngành Khoa học máy tính tại Khoa Công nghệ thông tin, trường Đại học Khoa học, Đại học Huế. Hiện đang là Giảng viên Trường Đại học Sư phạm Kỹ thuật, Đại học Đà Nẵng.

Lĩnh vực nghiên cứu: mạng truyền thông, mạng cảm biến không dây.

Đặng Đức Nam sinh năm 1979 tại Đà Nẵng. Ông tốt nghiệp Cử nhân Công nghệ Thông tin tại Đại học Khoa học Tự nhiên Hà Nội năm 2001. Ông là lập trình viên tại Softech Đà Nẵng từ năm 2001 đến năm 2007; Quản lý tại Unitech từ năm 2007 đến năm 2013; Quản lý cấp cao tại Axon Active Việt Nam – Chi nhánh Đà Nẵng từ năm 2013 đến nay.

Lĩnh vực nghiên cứu: công nghệ phần mềm, công nghệ IoT.

Phạm Trung Đức sinh năm 1988 tại Thừa Thiên Huế. Ông tốt nghiệp Cử nhân Công nghệ thông tin tại Trường Đại học Khoa học, Đại học Huế vào năm 2010; Nhận bằng Thạc sỹ chuyên nghành Khoa học máy tính tại Trường Đại học Khoa học, Đại học Huế năm 2012. Hiện nay là nghiên cứu sinh tại Trường Đại học Khoa học, Đại học Huế.

Lĩnh vực nghiên cứu: Mạng chuyển mạch chùm quang OBS, đa dạng dịch vụ QoS, điều khiển chấp nhận.

Dương Phước Đạt sinh năm 1987 tại Thừa Thiên Huế. Ông nhận bằng thạc sĩ Khoa học máy tính tại Trường Đại học Khoa học, Đại học Huế năm 2012, đang là nghiên cứu sinh ngành Khoa học máy tính tại Khoa Công nghệ thông tin, trường Đại học Khoa học, Đại học Huế. Hiện đang công tác tại Phòng Đào tạo, Khoa Du lịch – Đại học Huế.

Lĩnh vực nghiên cứu: mạng truyền thông, mạng OBS.