

MỤC LỤC

	Trang
LỜI MỞ ĐẦU.....	0
CHƯƠNG I. TỰ ĐỘNG KIỂM THỬ PHẦN MỀM.....	3
1. Kiểm thử phần mềm.....	3
1.2. Kiểm thử tích hợp – Intergration Test.....	5
1.3. Kiểm thử hệ thống – System test.....	6
1.4. Kiểm thử chấp nhận sản phẩm – Acceptance Test.....	7
2. Kiểm thử tự động (Automation test).....	8
2.1. Tại sao phải kiểm thử tự động ?.....	8
2.2. Các bước trong quy trình kiểm thử tự động.....	9
2.3. Một số thuận lợi và khó khăn khi áp dụng kiểm thử tự động vào kiểm tra phần mềm.....	10
CHƯƠNG II: TÌM HIỂU PHẦN MỀM LOADRUNNER.....	11
1. Giới thiệu phần mềm Loadrunner.....	11
2. Cài đặt phần mềm Loadrunner.....	12
3. Sử dụng phần mềm Loadrunner.....	12
3.1. Tạo kịch bản bằng Loadrunner.....	12
3.2. Hiệu chỉnh kịch bản.....	17
3.3. Tạo một Scenario Load Test.....	28
3.4. Phân tích Scenario.....	40
CHƯƠNG III. ỨNG DỤNG PHẦN MỀM LOADRUNNER KIỂM TRA HIỆU NĂNG WEBSITE.....	42
1. Giới thiệu.....	42
2. Tình huống kiểm thử.....	42
3. Phân tích kết quả.....	46
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	49
1. Kết luận.....	49
2. Hướng phát triển.....	49
TÀI LIỆU THAM KHẢO.....	50

LỜI MỞ ĐẦU

Với sự phát triển như vũ bão của công nghệ thông tin nói chung và công nghệ phần mềm nói riêng, việc phát triển phần mềm ngày càng được hỗ trợ bởi nhiều công cụ tiên tiến, giúp cho việc xây dựng phần mềm đỡ mệt nhọc và hiệu quả hơn. Tuy nhiên, vì độ phức tạp của phần mềm và những giới hạn về thời gian và chi phí, cho dù các hoạt động đảm bảo chất lượng phần mềm nói chung và kiểm thử nói riêng ngày càng chặt chẽ và khoa học, vẫn không đảm bảo được rằng các sản phẩm phần mềm đang được ứng dụng không có lỗi. Lỗi vẫn luôn tiềm ẩn trong mọi sản phẩm phần mềm và cũng có thể gây những thiệt hại khôn lường.

Kiểm thử phần mềm là một quá trình liên tục, xuyên suốt mọi giai đoạn phát triển phần mềm để đảm bảo rằng phần mềm thoả mãn các yêu cầu thiết kế và các yêu cầu đó đáp ứng các nhu cầu của người dùng. Các kỹ thuật kiểm thử phần mềm đã, đang được nghiên cứu, và việc kiểm thử phần mềm đã trở thành qui trình bắt buộc trong các dự án phát triển phần mềm trên thế giới. Kiểm thử phần mềm là một hoạt động rất tốn kém, mất thời gian, và khó phát hiện được hết lỗi.

Ngày nay tự động hóa được ứng dụng ở rất nhiều lĩnh vực, mục đích thường rất đa dạng và tùy theo nhu cầu đặc thù của từng lĩnh vực, tuy nhiên điểm chung nhất vẫn là giảm nhân lực, thời gian và sai sót. Ngành CNTT mà cụ thể là phát triển phần mềm cũng không ngoại lệ. Như chúng ta biết, để tạo ra sản phẩm CNTT hay phần mềm có chất lượng thì hoạt động kiểm tra phần mềm đóng vai trò rất quan trọng, trong khi đó hoạt động này lại tiêu tốn và chiếm tỷ trọng khá lớn công sức và thời gian trong một dự án. Do vậy, nhu cầu tự động hoá qui trình kiểm tra phần mềm cũng được đặt ra.

Áp dụng kiểm tra tự động hợp lý sẽ mang lại thành công cho hoạt động kiểm tra phần mềm. Kiểm thử tự động giúp giảm bớt công sức thực hiện, tăng độ tin cậy, giảm sự nhầm lẫn và rèn luyện kỹ năng lập trình cho kiểm tra viên.

Ở Việt Nam, trong thời gian qua việc kiểm thử phần mềm bị xem nhẹ, với công cụ lập trình hiện đại, người ta cảm tính cho rằng không kiểm thử cũng không sao, nên chưa có nhiều sự quan tâm, nghiên cứu. Những năm gần đây, một số tổ chức nghiên cứu và phát triển phần mềm đã bắt đầu có những quan tâm hơn đến vấn đề kiểm thử phần mềm. Tuy nhiên, vấn đề kiểm thử phần mềm hầu như vẫn chưa được đầu tư và quan tâm đúng mức. Nước ta đang trong quá trình xây dựng một ngành công nghiệp phần mềm thì không thể xem nhẹ việc kiểm thử phần mềm vì xác suất thất bại sẽ rất cao, hơn nữa, hầu hết các công ty phần mềm có uy tín đều đặt ra yêu cầu nghiêm ngặt

là nếu một phần mềm không có tài liệu kiểm thử đi kèm thì sẽ không được chấp nhận.

Qua những tìm hiểu về việc kiểm tra hiệu năng phần mềm, chúng ta có thể thấy được tầm quan trọng và vai trò của công việc này trong quy trình phát triển phần mềm, nhất là đối với những phần mềm ứng dụng lớn, có nhiều người sử dụng cùng một thời điểm như những ứng dụng Website, phần mềm quản lý tài chính, ngân hàng... Chính vì thế, em đã chọn đề tài: **“Tìm hiểu phần mềm Loadrunner kiểm tra hiệu năng WebSite”**.

Mục đích chính của đề tài:

- Tìm hiểu về quy trình kiểm tra chất lượng phần mềm và tự động kiểm tra phần mềm.
- Tìm hiểu về cách sử dụng phần mềm mã nguồn mở Loadrunner.
- Xây dựng kịch bản và tiến hành kiểm tra hiệu năng cho Website <http://student.vinhuni.edu.vn/CMCSoft.IU.Web.info/login.aspx>.

Đề tài bao gồm những nội dung sau:

- **Lời mở đầu.**
- **Chương I:** Tự động kiểm tra phần mềm.
- **Chương II:** Sử dụng phần mềm Loadrunner.
- **Chương III:** Ứng dụng phần mềm Loadrunner trong kiểm tra hiệu năng Website.
- **Kết luận và hướng phát triển.**

Trong khuôn khổ một đồ án tốt nghiệp, em chỉ tìm hiểu một cách tổng quan về tự động kiểm tra phần mềm (Automation Testing), các bước trong quy trình kiểm tra phần mềm và giới thiệu phần mềm Loadrunner để thực thi các kịch bản trong quá trình kiểm thử hiệu năng phần mềm. Cụ thể là kiểm tra hiệu năng của WebSite đăng ký học tín chỉ của trường Đại học Vinh. Tuy nhiên với kiến thức còn hạn chế của bản thân và không có nhiều tài liệu tiếng Việt về lĩnh vực này (hầu hết các tài liệu đều ở dạng tiếng Anh) nên đề tài chưa nghiên cứu được hết các khía cạnh trong qui trình kiểm tra phần mềm và phần mềm Loadrunner. Em hy vọng sẽ nhận được những ý kiến đóng góp quý báu từ phía các thầy cô và bạn bè để đề tài được hoàn thiện hơn. Hy vọng Automation Test sẽ phát triển mạnh trong tương lai ở Việt Nam, góp phần mang lại cho nền công nghệ phần mềm nước nhà những sản phẩm phần mềm đạt tiêu chuẩn và đáp ứng được nhu cầu ứng dụng công nghệ thông tin vào cuộc sống.

CHƯƠNG I
TỰ ĐỘNG KIỂM THỬ PHẦN MỀM

1. Kiểm thử phần mềm

Một chương trình mới được tạo ra thường chứa vài lỗi trong 100 dòng bao gồm lỗi từ quá trình lập trình và lỗi từ quá trình thiết kế. Nếu 1 chương trình chứa lỗi được dùng để vận hành 1 hệ thống trực tuyến, thì những hư hỏng nghiêm trọng phát sinh ra không chỉ ảnh hưởng tới công ty vận hành hệ thống đó mà còn ảnh hưởng cả tới công chúng lớn bên ngoài.

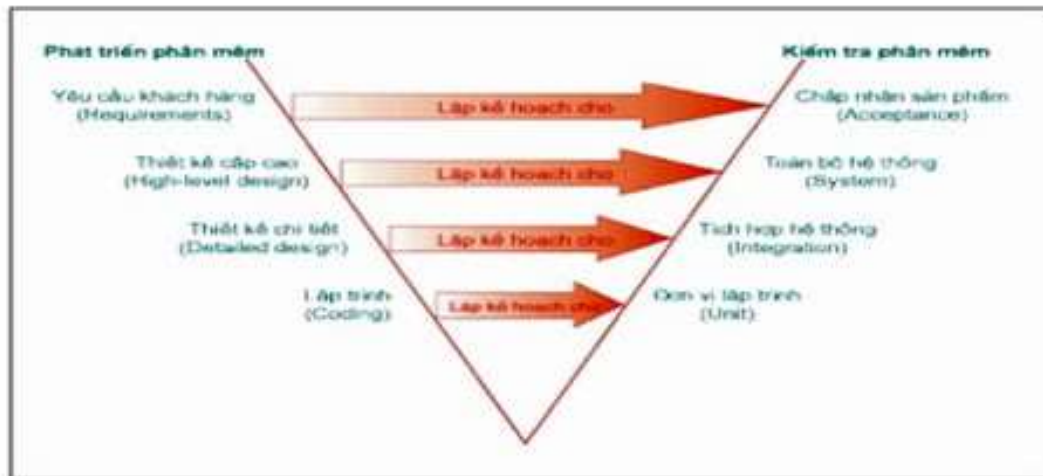
Do đó việc kiểm thử phần mềm phải được tiến hành trước khi chuyển giao sản phẩm công nghiệp. Việc kiểm thử phần mềm cũng phải được tiến hành theo 1 trình tự kiểm thử đặc biệt để kiểm chứng rằng chương trình và hệ thống mà nó điều khiển có thể vận hành tương ứng với các đặc tả. Mặc dù chúng ta không thể đảm bảo loại bỏ hoàn toàn hết lỗi trong chương trình nhưng chúng vẫn có thể làm giảm số lỗi đó tới mức tối thiểu nhất nếu chúng ta kiểm thử chương trình theo cách chính xác, hiệu quả.

Một hoạt động mang tính sống còn trong các dự án sản xuất hoặc gia công phần mềm, đó là kiểm tra (Testing). Người làm phần mềm chắc hẳn không ai nghi ngờ vai trò quan trọng của nó, tuy nhiên không phải ai cũng hiểu rõ hoạt động này. Bản thân công việc kiểm thử phần mềm cũng là một lĩnh vực hoạt động độc lập và khá “hấp dẫn”. Cùng với các dự án gia công sản xuất phần mềm, hiện cũng có khá nhiều dự án mà nội dung công việc chỉ là kiểm tra những phần mềm đã được khách hàng phát triển sẵn.

Thực tế cho thấy kiểm thử phần mềm là công việc mà bất cứ người nào từng tham gia phát triển phần mềm đều biết và từng làm. Kiểm thử phần mềm bao gồm việc “chạy thử” phần mềm hay một chức năng của phần mềm, xem nó “chạy” đúng như mong muốn hay không. Việc kiểm tra này có thể thực hiện từng chặng, sau mỗi chức năng hoặc module được phát triển, hoặc thực hiện sau cùng, khi phần mềm đã được phát triển hoàn tất.

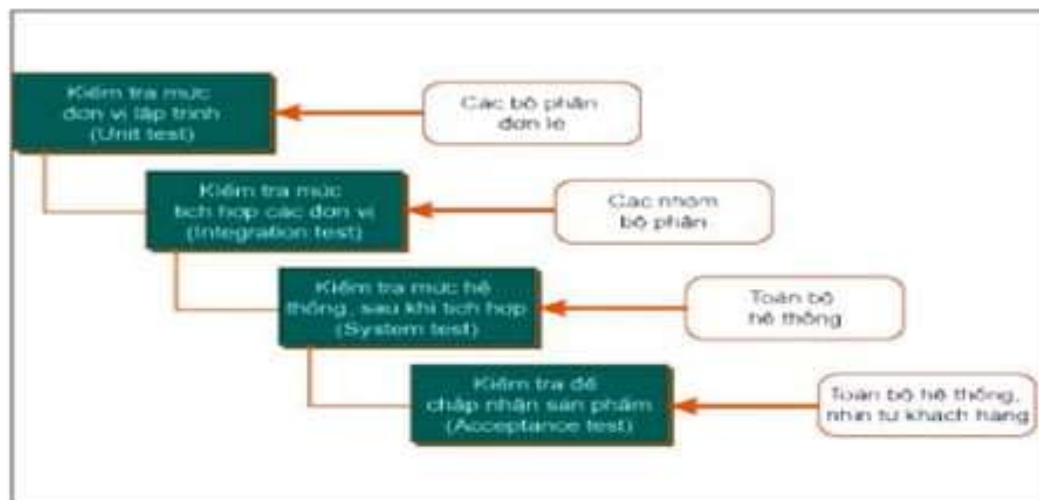
Do đó kiểm thử một sản phẩm phần mềm là xây dựng một cách có chủ đích những tập dữ liệu và dãy thao tác nhằm đánh giá một số hoặc toàn bộ các tiêu chuẩn của sản phẩm phần mềm đó.

Kiểm tra phần mềm có nhiều mức độ khác nhau và có mối tương quan với các chặng phát triển trong dự án phát triển phần mềm.



Hình 1.1 Tương quan giữa các chặng trong PTPM và KTPM

Kiểm tra phần mềm nói chung có 4 mức độ sau đây:



Hình 1.2 Các mức độ cơ bản của kiểm thử phần mềm

Kiểm thử mức đơn vị (Unit Test)

Một đơn vị là một thành phần phần mềm nhỏ nhất mà ta có thể kiểm thử được. Ví dụ, các hàm (*Function*), thủ tục (*Procedure*), lớp (*Class*) hay phương thức (*Method*) đều có thể được xem là Unit.

Unit Test thường do lập trình viên thực hiện. Công đoạn này cần được thực hiện càng sớm càng tốt trong giai đoạn viết code và xuyên suốt chu kỳ phát triển phần mềm. Unit Test đòi hỏi kiểm thử viên có kiến thức về thiết kế và code của chương trình. Mục đích của Unit Test là bảo đảm thông tin được xử lý và xuất là chính xác, trong mối tương quan với dữ liệu nhập và chức năng của Unit. Tất cả các nhánh bên trong Unit đều phải được kiểm tra để phát hiện nhánh phát sinh lỗi. Một nhánh thường

là một chuỗi các lệnh được thực thi trong một Unit. Ví dụ: chuỗi các lệnh sau điều kiện If và nằm giữa then ... else là một nhánh. Việc chọn lựa các nhánh để đơn giản hóa việc kiểm thử và quét hết Unit đòi hỏi phải có kỹ thuật, đôi khi phải dùng thuật toán để chọn lựa.

Cùng với các mục kiểm thử khác, Unit Test cũng đòi hỏi phải chuẩn bị trước các ca kiểm thử (*Test case*) hoặc kịch bản kiểm thử (*Test script*), trong đó chỉ định rõ dữ liệu đầu vào, các bước thực hiện và dữ liệu đầu ra mong muốn. Các Test case và Test script này nên được giữ lại để tái sử dụng.

Kiểm thử tích hợp – Intergration Test

Integration test kết hợp các thành phần của một ứng dụng và kiểm thử như một ứng dụng đã hoàn thành. Trong khi Unit Test kiểm tra các thành phần và Unit riêng lẻ thì Intgration Test kết hợp chúng lại với nhau và kiểm tra sự giao tiếp giữa chúng.

Hai mục tiêu chính của Integration Test:

- Phát hiện lỗi giao tiếp xảy ra giữa các Unit.
- Tích hợp các Unit đơn lẻ thành các hệ thống nhỏ (Subsystem) và cuối cùng là nguyên hệ thống hoàn chỉnh (System) chuẩn bị cho kiểm thử ở mức hệ thống (System Test).

Integration Test chỉ nên thực hiện trên những Unit đã được kiểm tra cẩn thận trước đó bằng Unit Test, và tất cả các lỗi mức Unit đã được sửa chữa. Một số người hiểu sai rằng Unit một khi đã qua giai đoạn Unit Test với các giao tiếp giả lập thì không cần phải thực hiện Integration Test nữa.

Một chiến lược cần quan tâm trong Integration Test là nên tích hợp dần từng Unit. Một Unit tại một thời điểm được tích hợp vào một nhóm các Unit khác đã tích hợp trước đó và đã hoàn tất các đợt Integration Test trước đó. Lúc này, ta chỉ cần kiểm thử giao tiếp của Unit mới thêm vào với hệ thống các Unit đã tích hợp trước đó, điều này sẽ làm cho số lượng can kiểm thử giảm đi rất nhiều, và sai sót sẽ giảm đáng kể.

Có 4 loại kiểm thử trong Integration Test:

Kiểm thử cấu trúc (Structure Test): Kiểm thử cấu trúc nhằm bảo đảm các thành phần bên trong của một chương trình chạy đúng và chú trọng đến hoạt động của các thành phần cấu trúc nội tại của chương trình.

Kiểm thử chức năng (Functional Test): Kiểm thử chức năng chỉ chú trọng đến chức năng của chương trình, không quan tâm đến cấu trúc bên trong, chỉ khảo sát chức năng của chương trình theo yêu cầu kỹ thuật.

Kiểm thử hiệu năng (Performance Test): Kiểm thử việc vận hành của hệ thống.

Kiểm thử khả năng chịu tải (Stress Test): Kiểm thử các giới hạn của hệ thống.

Kiểm thử hệ thống – System test

Mục đích System Test là kiểm thử thiết kế và toàn bộ hệ thống (sau khi tích hợp) có thỏa mãn yêu cầu đặt ra hay không.

System Test bắt đầu khi tất cả các bộ phận của phần mềm đã được tích hợp thành công. Loại kiểm thử này tốn rất nhiều công sức và thời gian. Trong nhiều trường hợp, việc kiểm thử đòi hỏi một số thiết bị phụ trợ, phần mềm hoặc phần cứng đặc thù, đặc biệt là các ứng dụng thời gian thực, hệ thống phân bố, hoặc hệ thống nhúng. Ở mức độ hệ thống, người kiểm thử cũng tìm kiếm các lỗi, nhưng trọng tâm là đánh giá về hoạt động, thao tác, sự tin cậy và các yêu cầu khác liên quan đến chất lượng của toàn hệ thống.

Điểm khác nhau then chốt giữa Integration Test và System Test là System Test chú trọng các hành vi và lỗi trên toàn hệ thống, còn Integration Test chú trọng sự giao tiếp giữa các đơn thể hoặc đối tượng khi chúng làm việc cùng nhau. Ta phải thực hiện Unit Test và Integration Test để bảo đảm mọi Unit và sự tương tác giữa chúng hoạt động chính xác trước khi thực hiện System Test.

Sau khi hoàn thành Integration Test, một hệ thống phần mềm đã được hình thành cùng với các thành phần đã được kiểm tra đầy đủ. Tại thời điểm này, lập trình viên hoặc kiểm thử viên bắt đầu kiểm thử phần mềm như một hệ thống hoàn chỉnh. Việc lập kế hoạch cho System Test nên bắt đầu từ giai đoạn hình thành và phân tích các yêu cầu.

System Test kiểm thử cả các hành vi chức năng của phần mềm và các yêu cầu về chất lượng như độ tin cậy, tính tiện lợi khi sử dụng, hiệu năng và bảo mật. Mức kiểm thử này đặc biệt thích hợp cho việc phát hiện lỗi giao tiếp với phần mềm hoặc phần cứng bên ngoài, chẳng hạn các lỗi "tắc nghẽn" (deadlock) hoặc chiếm dụng bộ nhớ. Sau giai đoạn System Test, phần mềm thường đã sẵn sàng cho khách hàng hoặc người dùng cuối cùng kiểm thử chấp nhận sản phẩm (*Acceptance Test*) hoặc dùng thử (*Alpha/Beta Test*). System Test thường được thực hiện bởi một nhóm kiểm thử viên hoàn toàn độc lập với nhóm phát triển dự án. System Test gồm nhiều loại kiểm thử khác nhau, phổ biến nhất gồm:

Kiểm thử chức năng (Functional Test): Bảo đảm các hành vi của hệ thống thỏa mãn đúng yêu cầu thiết kế.

Kiểm thử hiệu năng (Performance Test): Bảo đảm tối ưu việc phân bố tài nguyên hệ thống (ví dụ bộ nhớ) nhằm đạt các chỉ tiêu như thời gian xử lý hay đáp ứng

câu truy vấn...

Kiểm thử khả năng chịu tải (Stress Test hay Load Test): Bảo đảm hệ thống vận hành đúng dưới áp lực cao (ví dụ nhiều người truy xuất cùng lúc). Stress Test tập trung vào các trạng thái tới hạn, các "điểm chết", các tình huống bất thường như đang giao dịch thì ngắt kết nối (xuất hiện nhiều trong kiểm tra thiết bị như POS, ATM...)...

Kiểm thử cấu hình (Configuration Test).

Kiểm thử bảo mật (Security Test): Bảo đảm tính toàn vẹn, bảo mật của dữ liệu và của hệ thống.

Kiểm thử khả năng phục hồi (Recovery Test): Bảo đảm hệ thống có khả năng khôi phục trạng thái ổn định trước đó trong tình huống mất tài nguyên hoặc dữ liệu; đặc biệt quan trọng đối với các hệ thống giao dịch như ngân hàng trực tuyến...

Nhìn từ quan điểm người dùng, các cấp độ kiểm thử trên rất quan trọng: Chúng bảo đảm hệ thống đủ khả năng làm việc trong môi trường thực Tùy yêu cầu và đặc trưng của từng hệ thống, tùy khả năng và thời gian cho phép của dự án, khi lập kế hoạch, người quản lý dự án sẽ quyết định áp dụng những loại kiểm thử nào.

Kiểm thử chấp nhận sản phẩm – Acceptance Test

Sau giai đoạn System Test là Acceptance Test, được khách hàng hoặc ủy quyền cho một nhóm thử ba thực hiện. Mục đích của Acceptance Test là để chứng minh phần mềm thỏa mãn tất cả yêu cầu của khách hàng và khách hàng chấp nhận sản phẩm.

Acceptance Test có ý nghĩa hết sức quan trọng. Mặc dù trong hầu hết mọi trường hợp, các phép kiểm thử của System Test và Acceptance Test gần như tương tự, nhưng bản chất và cách thức thực hiện lại rất khác biệt.

Đối với những sản phẩm dành bán rộng rãi trên thị trường, thông thường sẽ thông qua hai loại kiểm thử gọi là kiểm thử Alpha – **Alpha Test** và kiểm thử Beta – **Beta Test**. Alpha Test, người dùng kiểm thử phần mềm ngay tại nơi phát triển phần mềm, lập trình viên ghi nhận các lỗi hoặc phản hồi và lên kế hoạch sửa chữa. Beta Test, phần mềm được gửi tới cho người dùng để kiểm thử ngay trong môi trường thực, lỗi hoặc phản hồi cũng sẽ gửi ngược lại cho lập trình viên để sửa chữa.

Nếu khách hàng không quan tâm và không tham gia vào quá trình phát triển phần mềm thì kết quả Acceptance Test sẽ sai lệch rất lớn, mặc dù phần mềm đã trải qua tất cả các kiểm thử trước đó. Sự sai lệch này liên quan đến việc hiểu sai yêu cầu cũng như sự mong chờ của khách hàng.

Gắn liền với giai đoạn Acceptance Test thường là một nhóm những dịch vụ và tài liệu đi kèm, phổ biến như hướng dẫn cài đặt, sử dụng v.v... Tất cả tài liệu đi kèm phải

được cập nhật và kiểm thử chặt chẽ.

2. Kiểm thử tự động (Automation test)

Là một software program dùng để chạy một cách tự động thay thế các thao tác testing bằng tay.

Là một phần mềm

Tại sao phải kiểm thử tự động ?

Test Tool trong lĩnh vực phát triển phần mềm là công cụ giúp thực hiện việc kiểm tra phần mềm một cách tự động. Tuy nhiên không phải mọi việc kiểm tra đều có thể tự động hóa, Test Tool thường được sử dụng trong một số tình huống sau:

- Không đủ tài nguyên

Khi số lượng tình huống kiểm tra (test case) quá nhiều mà các kiểm thử viên không thể hoàn tất bằng tay trong thời gian cụ thể nào đó.

Ví dụ khi thực hiện kiểm tra chức năng của một website. Website này sẽ được kiểm tra với 6 môi trường gồm 3 trình duyệt và 2 hệ điều hành.

Tình huống này đòi hỏi số lần kiểm tra tăng lên và lặp lại 6 lần so với việc kiểm tra cho một môi trường cụ thể.

- Kiểm tra hồi quy

Trong quá trình phát triển phần mềm, nhóm lập trình thường đưa ra nhiều phiên bản liên tiếp để kiểm tra. Việc đưa ra các phiên bản phần mềm có thể là hàng ngày, mỗi phiên bản bao gồm những tính năng mới, hoặc tính năng cũ được sửa lỗi, nâng cấp. Việc bổ sung hoặc sửa lỗi code ở phiên bản mới có thể làm cho những tính năng khác đã kiểm tra tốt chạy sai mặc dù phần code không hề chỉnh sửa. Để khắc phục, kiểm thử viên không chỉ kiểm tra chức năng mới hoặc được sửa, mà phải kiểm tra lại tất cả những tính năng đã kiểm tra tốt trước đó. Điều này khó khả thi về mặt thời gian nếu kiểm tra thủ công.

- Kiểm tra khả năng vận hành phần mềm trong môi trường đặc biệt

Nhằm đánh giá xem vận hành của phần mềm có thỏa mãn yêu cầu đặt ra hay không. Thông qua đó kiểm thử viên có thể xác định được các yếu tố về phần cứng, phần mềm ảnh hưởng đến khả năng vận hành của phần mềm. Một số tình huống kiểm tra tiêu biểu thuộc loại này:

- Đo tốc độ trung bình xử lý một yêu cầu của web server.
- Thiết lập 1000 yêu cầu, đồng thời gửi đến web server, kiểm tra tình huống 1000 người dùng truy xuất web cùng lúc.

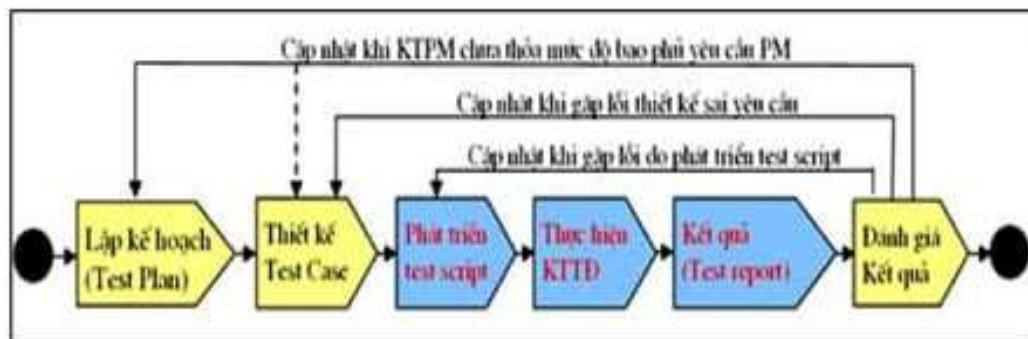
- Xác định số yêu cầu tối đa được xử lý bởi web server hoặc xác định cấu hình máy thấp nhất mà tốc độ xử lý của phần mềm vẫn hoạt động ở mức cho phép.

Việc kiểm tra thủ công cho những tình huống trên là cực khó, thậm chí “vô phương”.

Hoạt động kiểm thử tự động nhằm mục đích kiểm tra, phát hiện những lỗi của phần mềm trong những trường hợp đoán trước. Nó thường được thực hiện sau khi đã thiết kế xong các tình huống (test case). Tuy nhiên, không phải mọi trường hợp kiểm tra đều có thể hoặc cần thiết phải tự động hóa, trong tất cả test case thì kiểm thử viên phải đánh giá và chọn ra những test case nào phù hợp hoặc cần thiết để áp dụng kiểm thử tự động dựa trên những tiêu chí đã đề cập bên trên.

Các bước trong quy trình kiểm thử tự động

Việc phát triển kiểm thử tự động cũng tuân theo các bước phát triển phần mềm, chúng ta phải xem việc phát triển kiểm thử tự động giống như phát triển một dự án.



Hình 1.4 Tương quan giữa kiểm thử tự động và toàn bộ chu trình kiểm tra phần mềm

Giống như phát triển phần mềm, để thành công trong kiểm thử tự động, chúng ta nên thực hiện các bước cơ bản sau:

- Thu thập các đặc tả yêu cầu hoặc test case; lựa chọn những phần cần thực hiện kiểm thử tự động.
- Phân tích và thiết kế mô hình phát triển kiểm thử tự động..
- Phát triển lệnh đặc tả (script) cho kiểm thử tự động..
- Kiểm tra và theo dõi lỗi trong script của kiểm thử tự động..

Bảng sau mô tả rõ hơn các bước thực hiện kiểm thử tự động.

STT	Bước thực hiện	Mô tả
1	Tạo test script	Giai đoạn này chúng ta sẽ dùng test tool để ghi lại các thao tác lên PM cần kiểm tra và tự động sinh ra test script.
2	Chỉnh sửa test script	Chỉnh sửa để test script thực hiện kiểm tra theo đúng yêu cầu đặt ra, cụ thể là làm theo test case cần thực hiện.
3	Chạy test script để KTĐ	Giám sát hoạt động kiểm tra PM của test script.
4	Đánh giá kết quả	Kiểm tra kết quả thông báo sau khi thực hiện KTĐ. Sau đó bổ sung, chỉnh sửa những sai sót.

Một số thuận lợi và khó khăn khi áp dụng kiểm thử tự động vào kiểm tra phần mềm

❖ **Thuận lợi**

- Kiểm thử phần mềm không cần sự can thiệp của kiểm thử viên
- Giảm chi phí khi thực hiện kiểm tra số lượng lớn test case hoặc test case lặp lại nhiều lần.
- Giảm lập tình huống khó có thể thực hiện bằng tay

❖ **Khó khăn**

- Mất chi phí tạo các script để thực hiện kiểm thử tự động.
- Tốn chi phí dành cho bảo trì các script.
- Đòi hỏi kiểm thử viên phải có kỹ năng tạo script kiểm thử tự động.
- Không áp dụng được trong việc tìm lỗi mới của phần mềm.

CHƯƠNG II

TÌM HIỂU PHẦN MỀM LOADRUNNER

1. Giới thiệu phần mềm Loadrunner

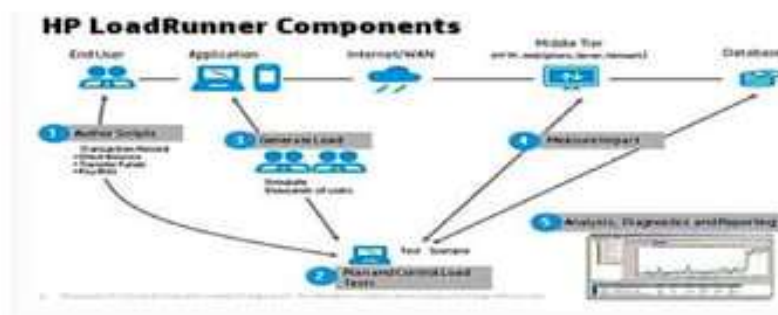
LoadRunner là công cụ kiểm thử tự động thực hiện việc kiểm tra hiệu năng của phần mềm. Nó cho phép chúng ta tìm ra những lỗi về khả năng thực thi bằng việc phát hiện nguyên nhân, chỗ làm cho phần mềm chạy chậm hoặc không đúng yêu cầu. Đây là công cụ mạnh với giải pháp kiểm tra tải, phát hiện và đưa ra giải pháp cải tiến.

Ứng dụng LoadRunner sẽ giúp giảm thời gian viết test script đến 80% nhờ có chức năng tự động phát sinh script mô tả lại các tình huống muốn kiểm tra.

LoadRunner có khả năng tạo ra hàng ngàn người dùng ảo thực hiện các giao dịch cùng một lúc. Sau đó LoadRunner giám sát các thông số xử lý của phần mềm được kiểm tra. Kết quả thống kê sẽ được lưu lại và cho phép kiểm thử viên thực hiện phân tích.

Loadrunner gồm có các thành phần sau:

- **Virtual User Generator:** Tự động tạo ra VuGen script để lưu lại các thao tác người dùng tương tác lên phần mềm. VuGen script này còn được xem là hoạt động của một người ảo mà LoadRunner giả lập.
- **Controller:** Tổ chức, điều chỉnh, quản lý và giám sát hoạt động kiểm tra tải. Thành phần này có chức năng tạo ra những tình huống (Scenario) kiểm tra.
- **Load Generator:** Cho phép giả lập hàng ngàn người dùng, hoạt động của từng người được thực hiện theo Vugen script. Kết quả thực hiện được thông báo cho Controller.
- **Analysis:** Cung cấp việc xem, phân tích và so sánh kết quả.
- **Launcher:** Nơi tập trung tất cả các thành phần của Loadrunner cho người dùng.



Hình 2.1 Mô hình hoạt động của Loadrunner

3. Cài đặt phần mềm Loadrunner

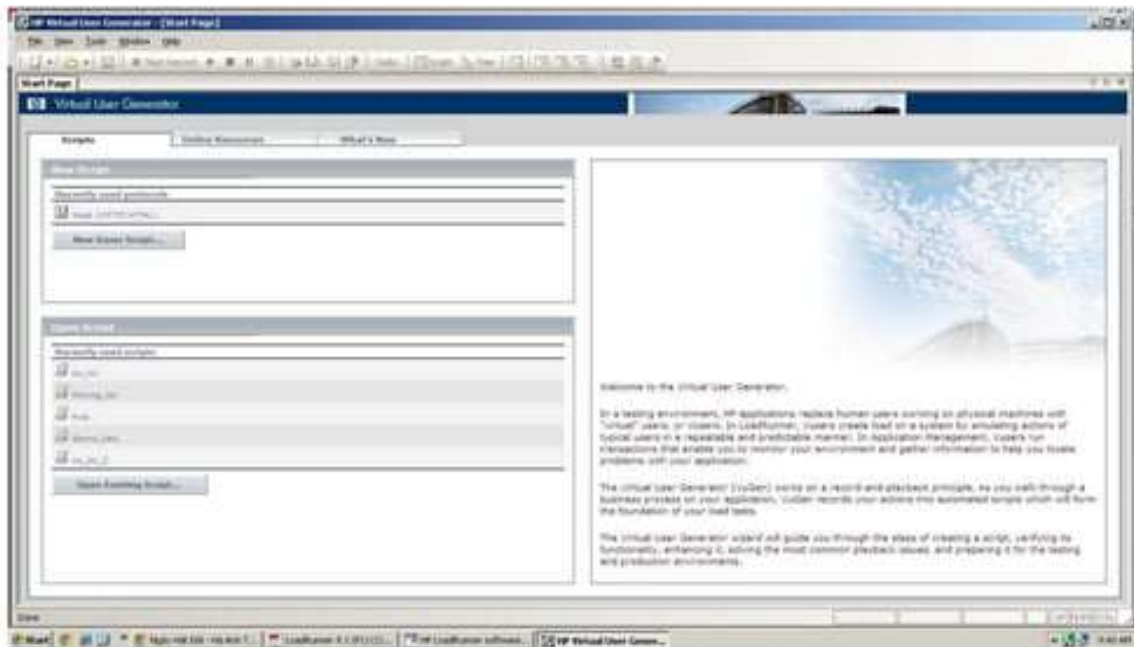
Yêu cầu	Thành phần			
	Controller	User Generator	Load Generator	Loadrunner
CPU	Pentium III trở lên (đề nghị Pentium IV) 1Ghz trở lên (đề nghị 2.4 Ghz)			
HĐH	Windows 2000 SP4 Windows 2003 PS3 (Standard and Enterprise editions) Window XP SP2 (đã tắt firewall)			
			HĐH dựa trên UNIX	
RAM	1GB	512MB trở lên (đề nghị 1GB)	Phụ thuộc giao thức và ứng dụng phải kiểm tra.	512MB trở lên (đề nghị 1GB)
Đĩa cứng	2GB	1GB	1GB	1GB
Trình duyệt	Internet Explorer 6.0 service pack 1 trở lên			

4. Sử dụng phần mềm Loadrunner**3.1. Tạo kịch bản bằng Loadrunner**

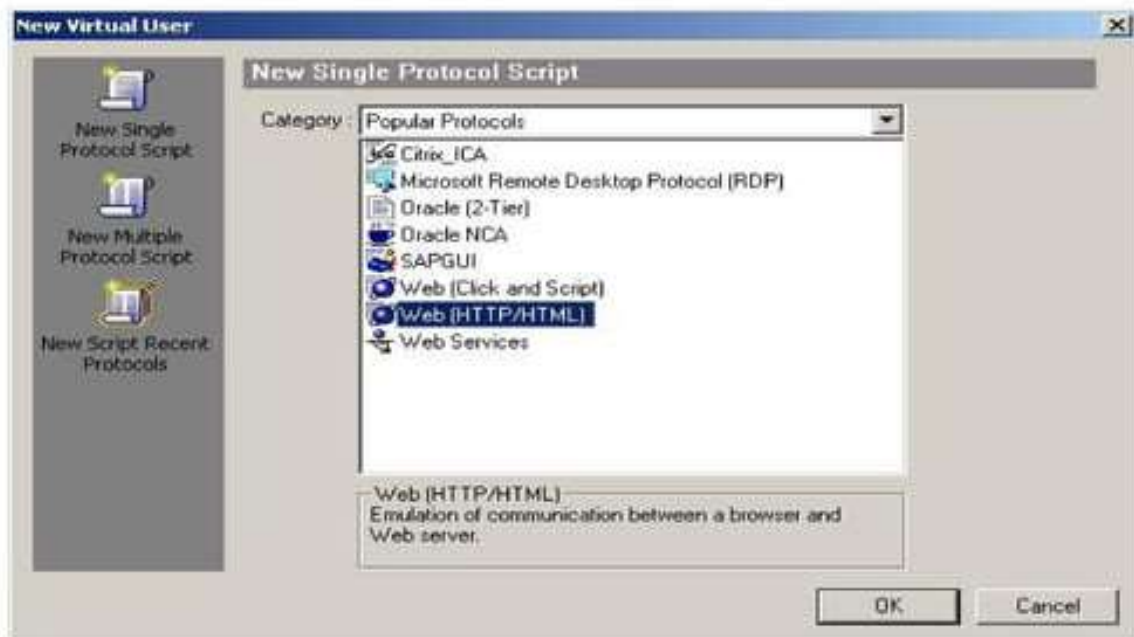
Kịch bản là những hành động mà người dùng sẽ tương tác với WebSite cần kiểm tra. Sau khi tạo kịch bản, ta phải đảm bảo kịch bản sẽ hoạt động bình thường giống như khi ta thực hiện đối với trang Web cần kiểm tra.


Trong một môi trường Test, LoadRunner thay thế con người sử dụng sức mạnh của máy tính, với người sử dụng ảo hay còn gọi là Vuser. Các Vuser tạo tải trên hệ thống mô phỏng hành động của người sử dụng và lặp đi lặp lại nhiều lần. VuGen hoạt động dựa trên nguyên tắc record-and-playback. Khi ta thực hiện các thao tác trên ứng dụng, VuGen tự động ghi lại các hành động vào trong script từ đó tạo thành nền tảng của Load Test.

Tại cửa sổ HP LoadRunner Launcher, chọn tab Load Testing, chọn Create/Edit Scripts. Trang VuGen's Start Page được hiển thị.



Tại VuGen's Start Page, chọn New Vuser Script trong tab Script. Hộp thoại New Virtual User mở ra, hiển thị những tùy chọn cho một script đơn thức mới.

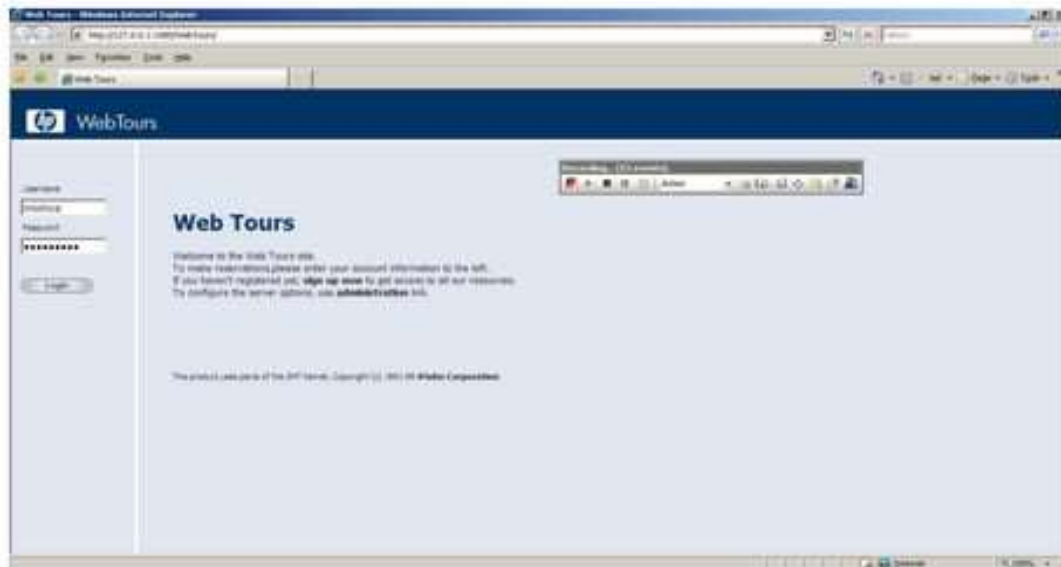


Nút ghi kịch bản  Start Record được sử dụng để lưu lại các hành động trong trình duyệt và ghi lại dưới dạng các đoạn mã JavaScript. Sau khi nhấn nút ghi kịch bản thì trình duyệt Web sẽ được tự động mở ra cho phép ta gõ địa chỉ của trang Web cần kiểm tra.



Giả sử ta sẽ kiểm tra trang <http://127.0.0.1:1080/WebTours/>. Đây là trang Web đặt vé máy bay trực tuyến. Sau khi bấm OK, một trình duyệt Web mới mở ra và hiển thị trang Web HP Tours đồng thời thanh công cụ Recording cũng được mở ra. Ta sẽ thực hiện một số thao tác trên trang Web này, các thao tác này sẽ được ghi lại để dùng cho việc đánh giá sau này.

Đầu tiên là thao tác Login



Hình 2.3. Trang Login

Màn hình thông báo quá trình Login đã thành công. Số events ở thanh Recording tăng lên theo hoạt động của Vuser.



Hình 2.4. Giao diện đăng nhập thành công

Ta lần lượt thực hiện các thao tác đặt vé máy bay, xem lại vé đã đặt, xem lại toàn bộ những vé đã từng đặt và thoát tài khoản.

Ta sẽ dừng ở đây và lưu lại các công việc đã thực hiện bằng cách nhấn vào nút Stop Recording.

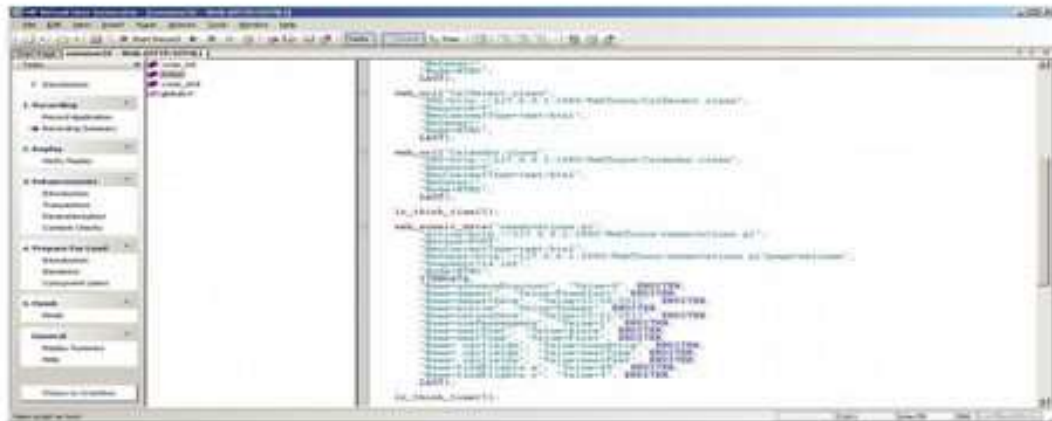
VuGen ghi lại từng bước từ lúc bạn bắt đầu click vào nút Start Record đến khi click vào nút Stop. Ta có thể xem lại script trong Tree View hoặc Script View.

Trong Tree View ta thấy được những hành động của người sử dụng theo từng bước. Hầu hết những bước này được kèm theo một bản chụp tương ứng trong quá trình ghi. Ta có thể so sánh các ảnh chụp sau đó để xác minh độ chính xác của script. VuGen cũng tạo ra ảnh chụp cho mỗi bước trong quá trình chạy lại.



Hình 2.6. Giao diện TreeView

Script view là một text-based liệt kê các hành động của Vuser bằng hàm API. Trong Script View, VuGen hiển thị script trong một chương trình biên soạn với màu sắc ứng với hàm và giá trị tham số. Bạn có thể nhập bằng ngôn ngữ C hoặc bằng hàm LoadRunner API, cũng như kiểm soát luồng xử lý rõ ràng và trực tiếp trong cửa sổ này.



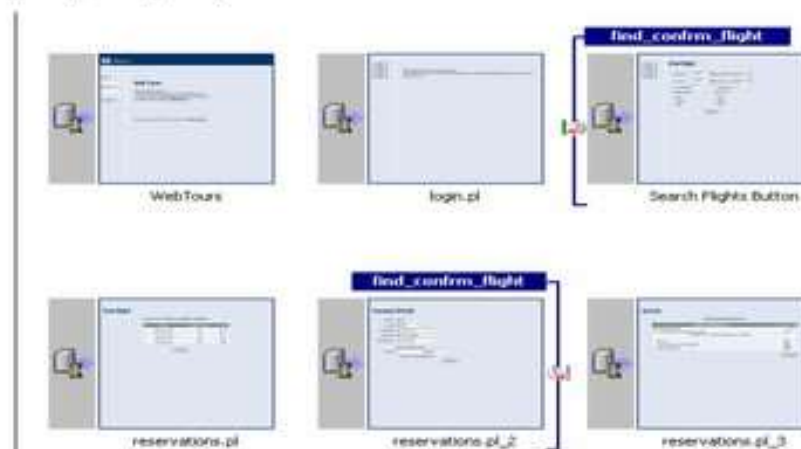
Hình 2.7. Giao diện Script view

Khi chuẩn bị cho việc triển khai một ứng dụng, cần đo lường khoảng thời gian của một quy trình nghiệp vụ cụ thể - mất bao lâu để đăng nhập, đặt chuyến bay ... Các quy trình nghiệp vụ này thường được thực hiện ở một hoặc nhiều bước hay nhiều hành động trong script. Trong LoadRunner, bạn chỉ định một loạt các hành động muốn đo lường bằng cách tạo các transaction.

LoadRunner thu thập thông tin về thời gian thực hiện một giao dịch và hiển thị các kết quả lên đồ thị và báo cáo theo từng mã màu. Ta sử dụng những thông tin này để xem ứng dụng có đáp ứng được những yêu cầu ban đầu hay không.

Để chèn một Transaction: Trong Task pane dưới tiêu đề Enhancements, click Transactions. Transaction Creation Wizard mở ra.

Transaction Creation Wizard hiển thị thumbnails của các bước khác nhau trong script. Click vào nút New Transaction để kéo các transaction đánh dấu điểm bắt đầu và kết thúc tương ứng trong script.



Sau khi đặt tên giao dịch và bấm OK, tên giao dịch sẽ được hiện ở menu bên phải. Và giao dịch cũng được chèn vào TreeView.



Đến đây ta ghi lại Project bằng cách chọn File/Save. Như vậy ta đã tạo được một kịch bản kiểm thử tuy nhiên kịch bản này chưa hoàn hảo, nó cần được chỉnh sửa.

3.2. Hiệu chỉnh kịch bản

Sau khi đã khởi tạo kịch bản thì công việc tiếp theo là phải kiểm tra lại kịch bản để xử lý các đoạn mã Script bị lỗi, có thể ảnh hưởng đến kết quả kiểm thử. Thỉnh thoảng, chạy lại một cách đơn giản sẽ không thành công, mặc dù quá trình ghi lại của cùng một hành động thì thành công. Nhiều ứng dụng sử dụng giá trị động để thay thế mỗi khi bạn sử dụng ứng dụng. Ví dụ như, một vài server được gán cho một sessionId duy nhất cho mỗi phiên làm việc mới. Khi bạn thử chạy lại một phiên làm việc đã được ghi lại, ứng dụng tạo ra một sessionId mới khác với sessionId đã được ghi lại. LoadRunner ghi địa chỉ vấn đề này thông qua sự tương quan. Sự tương quan này lưu lại những giá trị thay đổi, trong trường hợp sessionId, với một tham số. Khi chạy mô phỏng, các Vuser không sử dụng giá trị thay thế được ghi lại, nó sử dụng sessionId mới được gán từ server.

3.2.1. Tương quan SessionID

Để minh họa cho một playback thất bại, ta sửa đổi thiết định trong ứng dụng HP Tours. Thiết định này làm cho web server HP Tours không được phép tăng sessionId. Trong khi sửa đổi cấu hình HP Tours, máy chủ gán duy nhất một sessionId cho Vuser.

Mở HP Tours, thay đổi lựa chọn server: Click vào đường link administration trên trang HP Web Tours. Chọn checkbox thứ 3: **Set LOGIN form's action tag to an error page**. Xuống cuối trang và click update. Sau đó ta ghi lại một script mới và chạy lại bằng cách bấm vào Start Replay, hệ thống sẽ báo lỗi.



Ở ví dụ này, khi Loadrunner tự động thao tác Search_Flight_Button thì sẽ xảy ra lỗi về biến.



Để khắc phục vấn đề này, ta sử dụng VuGen tự động dò tìm khi cần để gắn sessionID. Sau đó khi chạy script, VuGen nhắc ta để quét script tương ứng.

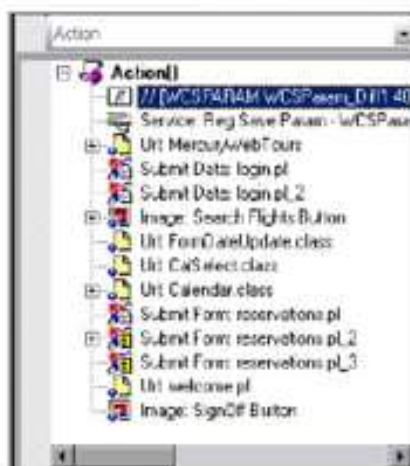
Ta sẽ hướng dẫn VuGen thêm vào một bước để lưu lại tham số sessionID gốc. Trong mỗi lần chạy lại, VuGen lưu lại sessionID mới cho tham số. Trong các bước sau, nó sử dụng giá trị đã lưu lại thay vì ghi lại giá trị ban đầu.

Click vào đường link Show và giải quyết những giá trị server động trong ô hướng dẫn dưới tiêu đề Dynamic Server Values. VuGen quét script, tìm kiếm sự khác biệt giữa các giá trị đã ghi lại và phát lại những giá trị đó. VuGen hiển thị một danh sách những khác biệt này để yêu cầu sự tương quan trong tab Correlation Result trong cửa sổ Output.



Chọn mục đầu tiên trong tab Correlation Results, và click Correlate.

VuGen thêm một bước mới vào trên cùng của script, nhằm lưu thông tin của tham số sessionId gốc. Trong mỗi lần chạy lại, VuGen lưu tham số sessionId mới. Trong các bước sau, sử dụng giá trị đã lưu thay vì ghi lại giá trị ban đầu.



Khi VuGen thêm vào script thì được trình bày như sau:

```
Web_reg_save_param ("WCSParam_Diff1",
"LB=userSession value=",
"RB=>", "Ord=1",
"RelFrameId=1.2.1",
"Search=Body",
LAST);
```

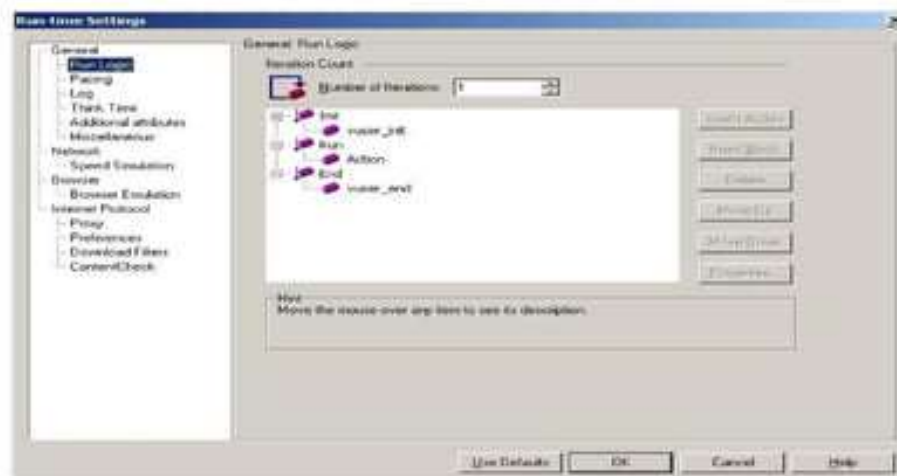
Có nghĩa là “Kiểm tra sự đáp ứng từ phía server cho dữ liệu được đặt giữa hai chuỗi”. Giá trị bên trái userSession value= và giá trị bên phải >. Lưu sự kiện xảy ra đầu tiên của dữ liệu này vào một tham số được gọi là **WCSParam_Diff1**.

Nghĩa là lúc này biến sessionId sẽ được tự động cập nhật và không có lỗi nữa.



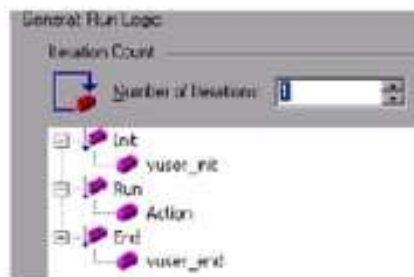
3.2.2. Thiết lập thời gian chạy

Cài đặt thời gian chạy trong LoadRunner giúp ta mô phỏng những tính chất khác nhau của hoạt động và hành vi của người sử dụng thực. Ví dụ, có thể mô phỏng một người sử dụng đáp ứng ngay lập tức thông tin từ server, hay một người dùng dừng lại trước mỗi phản hồi. Ta cũng có thể cấu hình cài đặt thời gian chạy để xác định có bao nhiêu lần Vuser nên lặp lại các hành động và thường xuyên như thế nào. Hiện có cài đặt thời gian chạy chung và cài đặt cụ thể cho một số loại Vuser. Ví dụ như một Website mô phỏng, ta có thể cho các Vuser chạy trong Netscape thay vì trong Internet Explorer..



Hình 3.2. Hộp thoại Run Time Setting

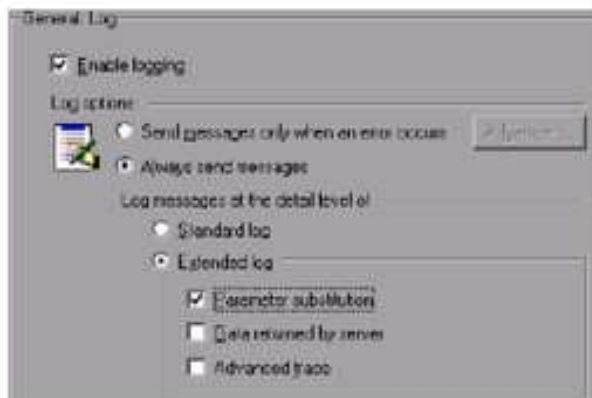
Mở cài đặt Run Logic: Thiết lập số lần lặp lại, hay số lần lặp lại các hành động trong chuỗi. Thiết lập số lần lặp đến 2.



Thiết lập cấu hình Pacing: Nút này cho phép ta kiểm soát thời gian giữa các lần lặp. Ta sẽ chỉ định một thời gian ngẫu nhiên. Những mô phỏng chính xác một cấu hình real-life nơi mà người sử dụng chờ giữa các hành động, nhưng tại những khoảng thời gian ngẫu nhiên không thể thấy được người sử dụng thực chờ chính xác 60 giây giữa các lần lặp. Chọn option thứ ba và chọn các thông tin sau: At Random intervals, every 60.000 to 90.000 sec.

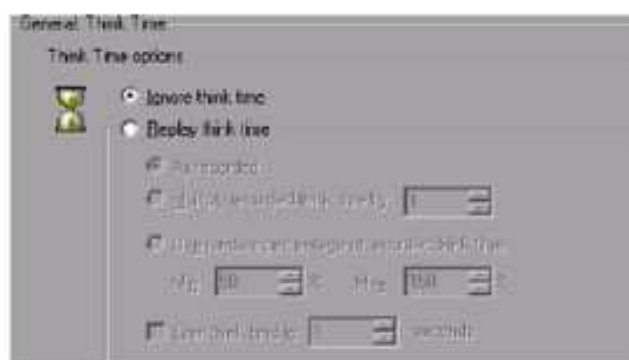


Thiết lập cài đặt Log: Cài đặt Log để ghi lại thông tin trong quá trình chạy kiểm tra. Trong quá trình triển khai, ta có thể chọn Enable logging cho mục đích debug, nhưng một khi đã xác định được script là function, ta có thể Enable logging chỉ để tìm lỗi hoặc Disable nó. Chọn Extended log và hiển thị Parameter Substitution.



Xem cài đặt thời gian: Không thay đổi để khi chạy script trong VuGen, nó sẽ chạy nhanh chóng vì nó không bao gồm Think-Time. Nếu cần thì sẽ thiết lập Think-

Time từ Controller.

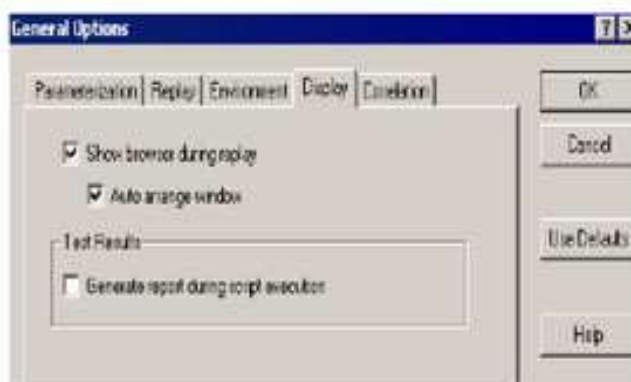


3.2.3. Xem script đang chạy trong thời gian thực

Chức năng xem Run-Time của Vugen hiển thị hoạt động của Vuser trong thời gian thực khi ghi và chạy lại script.

Mặc định, VuGen chạy ngầm mà không hiển thị một hình ảnh nào của hành động trong script. Ta sẽ định hướng VuGen hiển thị những hành động trong đó để có thể xem VuGen thực thi ở mỗi bước như thế nào thông qua các ảnh chụp của từng trang.

- Chọn Tool / General Option và chọn tab Display.
- Chọn Show browser during replay và check vào Auto arrange Window.

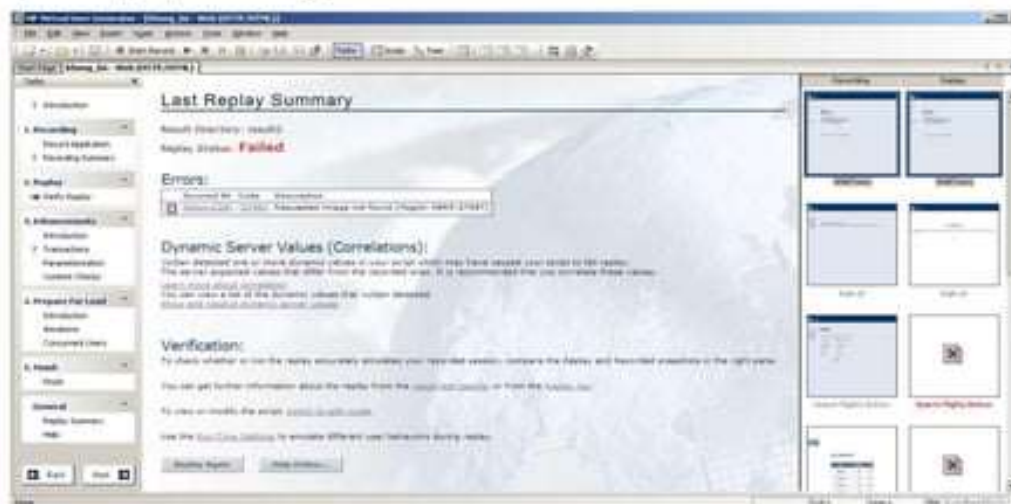


- Click OK để đóng hộp thoại General Options.
- Click Verify Replay trong Task pane và sau đó click nút Start Replay. Ngoài ra có thể bấm phím F5 hoặc click vào nút Run trong thanh công cụ.
- Nếu hộp thoại Select Results Directory mở và hỏi muốn lưu kết quả ở đâu, chấp nhận với tên mặc định thì click OK.

Sau một vài phút, VuGen mở ra một Run-Time viewer và bắt đầu chạy script trong Script view hoặc Tree view, tùy thuộc vào ctab đã mở cuối cùng. Trong Run-Time viewer, ta có thể quan sát những hành động của Vuser một cách trực quan.

3.2.4. Xem thông tin chạy lại

Khi script ngừng hoạt động, ta có thể xem thông tin tóm tắt của quá trình chạy lại. Click Verify Replay trong Task pane để xem thông tin tóm tắt của lần chạy lại mới nhất (Last Replay Summary).



Last Replay Summary liệt kê danh sách các lỗi được phát hiện và hiển thị thumbnails của những hình ảnh được chụp lại trong quá trình ghi và phát lại.

Ta cũng có thể nhìn thấy hành động của Vuser bằng cách xem lại bảng tóm tắt của các sự kiện ở tab Replay Log như sau:

- Chọn đường link Replay log trong ô hướng dẫn. Hoặc click vào nút Show/Hide trong thanh công cụ hoặc chọn View / Output Window từ menu. Sau đó click vào tab Replay Log.
- Nhấn Ctrl+F trong Replay Log để mở hộp thoại Find.
- **Started, Terminated:** Bắt đầu và kết thúc của script (Virtual User Script Started, Vuser Terminated)
- **Iteration:** Bắt đầu và kết thúc của vòng lặp và số lần lặp (Chữ màu cam).

VuGen hiển thị những bước thành công bằng màu xanh (green) và hiển thị lỗi bằng màu đỏ (red). Ví dụ như: nếu kết nối bị ngắt trong khi thực hiện test, VuGen sẽ dòng lỗi và hiển thị toàn bộ dòng đó bằng chữ màu đỏ.

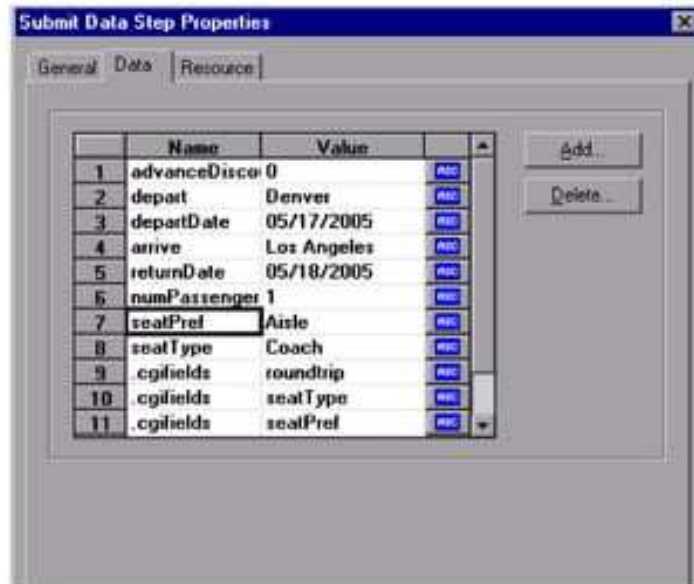
3.2.5. Mô phỏng nhiều người dùng

Trong ví dụ này, ta muốn theo dõi một người dùng đặt một chuyến bay và chọn một ghế ngồi. Tuy nhiên trong cuộc sống thực tế những người dùng khác nhau sẽ có những sở thích khác nhau. Để cải tiến việc test, ta sẽ kiểm tra xem việc đặt vé khi

người dùng chọn những sở thích chỗ ngồi khác nhau (lối đi, cửa sổ, không chọn).

Để thực hiện điều này, ta sẽ biểu hiện dưới dạng tham số các script. Ta sẽ đi ghi lại các giá trị và thay thế nó bằng một tham số. Ta sẽ lưu trữ giá trị cho tham số trong một tập tin tham số. Khi thực thi đoạn script, Vuser sẽ lấy giá trị từ tập tin tham số (lối đi, cửa sổ hoặc không chọn).

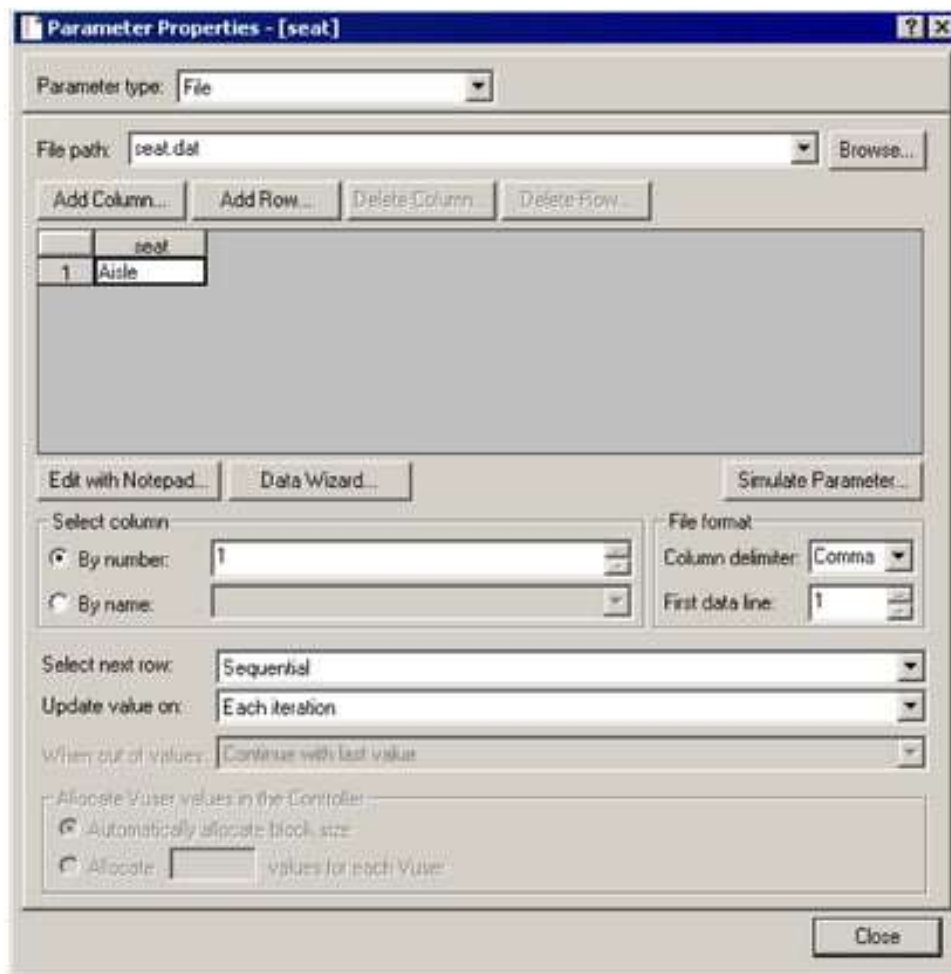
Trên Test Tree nhấn đúp vào bước **Submit Data: reservations.pl**. Hộp thoại về các thuộc tính của bước Submit Data được mở lên :



Chọn seatPref ở dòng thứ 7 có giá trị là Aisle. Click vào biểu tượng bên cạnh giá trị Aisle, hộp thoại Select or Create Parameter được mở ra:



Chỉ định tên tham số, chỗ ngồi, sử dụng tập tin loại tham số, click OK Vugen thay thế biểu tượng ABC với một biểu tượng tham số. Chọn Parameter Properties từ menu popup, hộp thoại Parameter Properties được mở lên.



Click AddRow. Vugen sẽ thêm một dòng vào trong bảng. Thay thế các từ Value bằng Window. Làm tương tự để thêm dòng None.

Giữ cài đặt mặc định trong hộp thoại Select column and File format.

Giữ cài đặt mặc định để hướng dẫn Vugen giữ giá trị liên tục cho mỗi sự lặp lại không phải là giá trị ngẫu nhiên.

Select next row: Random.

Update value on: Each iteration.

Đóng hộp thoại Parameter Properties và click OK để đóng hộp thoại Step Properties sau đó chạy lại script. Vì có 3 sự lựa chọn về chỗ ngồi (Cửa sổ, Lối đi, Không chọn) nên ta sẽ cài đặt cho kịch bản chạy ba vòng. Để thay đổi số vòng chạy của một script ta chỉnh ở cửa sổ Run – time Setting.

Ở trên ta chọn **Select next row:** Random nên kết quả sẽ ra ngẫu nhiên một trong 3 vị trí đó.


```

Action.c(24):      * name="roundtrip" value="on" />Roundtrip ticket</td></tr><tr><td>Seating Preference</td>
Action.c(24):      <td><td>Type of Seat</td></tr><tr><td>input type="radio" name="seatsPref" value="A1">
Action.c(24):      e" />Aisle</td><td>input type="radio" name="seatsPref" value="Window" />Window</td><td>input

```

Chạy lần 1

```
Action.c(58):      scount = value="0" /><input type="hidden" name="sestType" value="Coach" /><input type="hid
Action.c(58):      m: name="sestPri" value="None" />>>
Action.c(56):      <P><center><table width=80% cellpadding=1><tr><td align=center><div>
```

Chạy lần 2.

```
Action.c(83): en name="passengers" value="1" />input type="hidden" name="seatType" value="Coach" />
Action.c(83): <input type="hidden" name="departure" value="Rome" />input type="hidden" name="outboundFl
Action.c(83): nht" value="482:1950:12/02/2012" />input type="hidden" name="advancedDiscount" value="0"
```

Chạy lần 3.

3.2.6. Xác minh nội dung trang web

Khi thực hiện việc test, cần phải xác minh tính chính xác nội dung được tìm thấy ở trang trả về. Một nội dung kiểm tra phải xác minh thông tin mong đợi xuất hiện trên trang web khi script được thực thi. Ta có thể chèn 2 loại nội dung kiểm tra : text check và image check

- Text check kiểm tra một chuỗi văn bản xuất hiện trên trang web.
- Image check kiểm tra hình ảnh xuất hiện trên trang web.

Tim kiếm một văn bản. Ở ví dụ này ta sẽ thêm một đoạn văn bản kiểm tra để kiểm tra cụm từ "Find Flight" xuất hiện trên một vùng của trang trên script.

Chèn vào một đoạn văn bản kiểm tra : Mở Content Check Wizard, click Content Checks. Content Check wizard hiển thị thumbnail của mỗi bước trên script.



Chọn tab Page View bên phải bảng để hiển thị ảnh chụp của thumbnail.

Chọn trang chứa các văn bản muốn kiểm tra: Click vào thumbnail thứ tư
reservations.pl.

Chọn văn bản muốn kiểm tra: Đánh dấu các từ Find Flight trên ảnh chụp của thumbnail. Click chuột phải và chọn Add a Text Check (web-reg-find). Hộp thoại Find