Log Playback Testing

Vijay Upadya
Microsoft

Agenda

- What is log playback testing?
- Why should it be used and when?
- How to play back tests?
- How to use it for reproducing bugs?
- How to use it for data driven testing?
- Illustration using real world example

Introduction

- Logs typically are
  - used for diagnosing issues reported by customers
  - analysis is done manually
- Taking it to next step
  - Use log file as source of information for automated repro
  - Use log file for enhancing the testing

Definition

- Playing back the steps of product usage scenario using log file as source of information

Why log playback - Motivation

- Real world logs
  - play back for repro
  - convert them to real world tests
- Internal logs
  - determine 'test' points
  - determine states and transitions for a given scenario (white box tests)
- Approach useful for client apps and server apps, but more so for apps that have non-deterministic control flows

How to playback

There are two main pieces of information needed for playback

- User intent (i.e., scenario user intended to accomplish)
- Path taken by product in fulfilling this intent (i.e., product's execution flow)

User Intent

- *User intent* is about capturing what a user intended to do while using the product. It could be a simple task or a series of tasks as part of a larger scenario

- Examples:
  - User performing mathematical operations on a series of numbers in a calculator application
  - User copying photos from camera to local folder on his PC and syncing it to other PCs using file synchronization software.

How to capture user intent

- This essentially means logging
  - User actions
  - Data used by those actions
  - Timeline info

Exercising product path

- *Path taken by the product*: When a user performs a specific task, the product itself will perform a series of tasks to accomplish the user intent. These tasks are typically implemented in multiple components and some of these could be executed in parallel.

- There are two items of interest for capturing product path
  - Timeline of states and their transitions while executing tasks
  - Information on any concurrent execution of tasks

How to capture product path

- Logging states and their transitions of various components as they get exercised
- Logging timeline info

Putting it all together

Example: File Sync Software

User actions:

Create , Update, Delete of files/folders

Product Control flow:

On sender machine: Scanning, Uploading,

On receiver machine: Downloading, Realizing

Log file snippet

- 06-14-2011 00:08:46.507 SCANNER: thread for 'C:\photos' started
- 06-14-2011 00:08:46.617 SCANNER: FILE_ACTION_ADDED for 'C:\photos\ myPhoto1.jpg'
- 06-14-2011 00:08:48.703 SCANNER: thread finished scanning
- ..
- 06-14-2011 00:08:49.826 DOWNLOADING file 'myPhoto2.jpg'
- 06-14-2011 00:08:49.827 UPLOADING file 'C:\photos\myPhoto1.jpg'
- ..
- 06-14-2011 00:08:51.626 DOWNLOADING file 'myPhoto2.jpg'
- 06-14-2011 00:08:51.629 UPLOADING file 'C:\photos\myPhoto1.jpg'
- ..
- 06-14-2011 00:08:53.322 DOWNLOADING file 'myPhoto2.jpg'
- 06-14-2011 00:08:53.327 UPLOADING file 'C:\photos\myPhoto1.jpg'

Interpreting log

- From logs we have following information:
  - User intent: Addition of file on two machines and syncing to all machines
  - Product path: Scanning completed, Simultaneous download and upload task started

Steps for playback

These are the steps playback tool need to perform for the example log:

- Create file myPhoto1.jpg in machine1
- Create file myPhoto2.jpg in machine2
- Wait for download to start for file myPhoto2.jpg
- As soon as download starts, also start uploading myPhoto1.jpg

Control flow playback

Detours: Detours library enables hijacking function calls in the product and gives test code a chance to control its execution timings. The way this is used in the example case for file synchronization is:

- Create a test binary that detours upload and download functions by providing a detour function
- Load this test binary to the product process
- In the upload detour function wait until download is ready to begin
- In the download detour function wait until upload is ready to begin
- When the condition is met, continue the execution

Test Generation

Interpret logs to find different possible states and exercise

In the example log there are 4 states :

Scanning, Uploading, Downloading, Realizing

Possible state combinations during download that's of interest for testing

- Download during Scanning
- Download during Uploading
- Download during Realizing

Test Generation (Cont.)

- Generate 3 different test cases to test these 3 different possible state combinations
- Use detours to control the product and force these state combinations for testing

Conclusion

- Requires initial investment
- Make repro of issues less painful
- Convert real world issues into regression tests
- Test different code paths around user scenario to improve code coverage