Introduction to Software Testing
(*2nd edition*)
Chapter 3

Test Automation

Paul Ammann & Jeff Offutt

http://www.cs.gmu.edu/~offutt/softwaretest/

Updated October 2018

What is Test Automation?

The use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions

استفاده از نرم افزار برای کنترل اجرای آزمونها، مقایسه نتایج واقعی با نتایج پیش بینی شده، تنظیم پیش شرطهای آزمون، و سایر عملکردهای کنترل و گزارش تست

- Reduces cost
- Reduces human error
- Reduces variance in test quality from different individuals
- Significantly reduces the cost of regression testing by allowing a test to be run repeatedly.

هزینه را کاهش می دهد
خطای انسانی را کاهش می دهد
واریانس کیفیت تست را از افراد مختلف کاهش می دهد
به طور قابل توجهی هزینه تست رگرسیون را با اجازه دادن به یک تست برای اجرای مکرر کاهش می دهد.

Software testing can be expensive and labor intensive, so an important goal of software testing is to automate as much as possible.

تست نرم افزار می تواند پرهزینه و کار فشرده باشد، بنابراین هدف مهم تست نرم افزار خودکارسازی تا حد امکان است.

Types of tasks

- *Revenue* tasks: contribute directly to the solution of a problem.
  - Example: determining which methods are appropriate to define a data abstraction in a Java class.

کمک مستقیم به حل یک مشکل.
کارهایی که نیاز به خلاقیت ذهن انسان داره مثلا میخاهیم کلاس دیاگرام های یک سیستم را طراحی کنیم . یا مثلا یه کلاس داریم میخایم تصمیم بگیریم چه متدهایی براش بنویسیم؟ یعنی مسائلی که باتوجه به تحلیل های مختلف میتواند جواب های متفاوتی داشته باشه. که کاملا با دخالت ذهن و خلاقیت انسان انجام میشه

- *Excise* tasks: do not.
  - Example: compiling a Java class, because contributes nothing to the behavior of that class.

نیاز به خلاقیت و مشارکت خاصی ندارند مثل کامپایل کردن کلاس ها فقط یه مجموعه قانون باید بررسی شه که اون کلاس طبق قوانین نوشته شده یا نه؟

Excise tasks are candidates for automation;

تسک هایی که به صورت اتوماتیک قابل انجام هستند و به دخالت انسان ها نیاز ندارند.

Types of tasks

- Software testing probably has more excise tasks than any other aspect of software development.

- Maintaining test scripts, rerunning tests, and comparing expected results with actual results are all common excise tasks that routinely use large amounts of test engineers' time.

کارهایی که به صورت تکراری و اتوماتیک میتوانند انجام شوند و به خلاقیت انسان نیاز ندارند.

نگهداری از اسکریپتهای تست، اجرای مجدد تستها و مقایسه نتایج مورد انتظار با نتایج واقعی، همگی از وظایف متداول هستند که به طور معمول زمان زیادی از مهندسان تست استفاده میکنند.

Benefits of automating excise tasks

سختی کار را کمتر میکنند و باعث لذت بخش تر شدن شغل مهندس تست میشوند.

- Eliminating excise tasks eliminates drudgery, thereby making the test engineer's job more satisfying.
- Automation frees up time to focus on the fun and challenging parts of testing, such as test design, a revenue task.

تمرکز مهندس تست بره سمت جاهایی که نیاز به خلاقیت داره مثل طراحی تست ها وتسک های revenue

- Automation allows the same test to be run thousands of times without extra effort in environments where tests are run daily or even hourly.

Benefits of automating excise tasks (Cnt'd)

- Automation can help eliminate errors of omission, such as failing to update all the relevant files with the new set of expected results.
- Automation eliminates some of the variance in test quality caused by differences in individual's abilities.

کاهش تنوع کیفیت تست یعنی دیگه به توانایی انسان ها وابسته نمیشه

ممکنه فراموش کنیم همه فایل ها رااپدیت کنیم ولی اگه اتومات باشه دیگه فراموش نمیشن یک بار فقط کدش را مینویسیم و دفعات بعد خودش پروسه را انجام میده

Software Testability (3.1)

- Estimates how likely testing will reveal a fault if one exists.
- Plainly speaking – how easy or hard it is for faults to escape detection in the software

اگه یه faultای وجود داره اون تست ما با چه درجه ای میتواند اشکارش کنه؟

تخمین میزند که در صورت وجود خطا، انجام تست چقدر احتمال دارد که خطا را آشکار کند.
به زبان ساده - چقدر آسان یا سخت است که خطاها در نرم افزار شناسایی نشوند

The degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met

Software Testability (3.1)

- Testability is dominated by two practical problems
  - How to provide the test values to the software
  - How to observe the results of test execution

دو مشکل عملی بر آزمون پذیری غالب است
– نحوه ارائه مقادیر تست به نرم افزار
– نحوه مشاهده نتایج اجرای آزمون

Observability and Controllability

- Observability

قابلیت مشاهده
مشاهده رفتار یک برنامه از نظر خروجی ها، اثرات آن بر محیط و سایر اجزای سخت افزاری و نرم افزاری چقدر آسان است.

How easy it is to observe the behavior of a program in terms of its outputs, effects on the environment and other hardware and software components

  - Software that affects hardware devices, databases, or remote files have low observability

نرم افزارهایی که بر دستگاه های سخت افزاری، پایگاه های داده یا فایل های راه دور تأثیر می گذارند، قابلیت مشاهده کمی دارند

- Controllability

How easy it is to provide a program with the needed inputs, in terms of values, operations, and behaviors

  - Easy to control software with inputs from keyboards
  - Inputs from hardware sensors or distributed software is harder

قابلیت کنترل
ارائه یک برنامه با ورودی های مورد نیاز، از نظر مقادیر، عملیات و رفتار چقدر آسان است
– کنترل آسان نرم افزار با ورودی از صفحه کلید
- ورودی از سنسورهای سخت افزاری یا نرم افزارهای توزیع شده سخت تر است

Observability and Controllability

- Data abstraction reduces controllability and observability

- Many observability and controllability problems can be addressed with *simulation*,
  - By extra software built to "bypass" the hardware or software components that interfere with testing

انتزاع داده ها قابلیت کنترل و مشاهده را کاهش می دهد
بسیاری از مشکلات قابل مشاهده و کنترل را می توان با شبیه سازی حل کرد،
- توسط نرم افزار اضافی ساخته شده برای "دور زدن" قطعات سخت افزاری یا نرم افزاری که در تست تداخل دارند

ر فرایند تست دخالت میکنیم تا اون observability , controllability را هش دست پیدا کنیم تا حدی

Types of software with low observability and controllability

- Embedded software,
- Component-based software,
- Distributed software,
- Web applications.

تست کردنشون سخت تره
ورودی ها رو مستقیم نمیشه بهشون داد.

اجرای یک نرم افزار به اجرای نرم افزار دیگه ای وابسته است.

یعنی فالت ها
درحدممکن باید
بتوانند خودشون را
نشون بدند.

Testability is crucial to test automation because test scripts need to control the execution of the component under test and to observe the results of the test.

تستپذیری برای تست اتوماسیون بسیار مهم است زیرا اسکریپتهای تست نیاز به کنترل اجرای کامپوننت تحت آزمایش و مشاهده نتایج تست دارند.

هرچه testability پایین باشه نمیتوانیم به طور شفاف فرایند تست را انجام بدیم و نتایج ای که میگیریم نمیتوانند معیار مناسبی برای مقایسه و کیفیت باشند.

Components of a Test Case (3.2)

- A test case is a multipart artifact with a definite structure

یک نمونه آزمایشی یک مصنوع چند بخشی با ساختار مشخص است

- Test case values

The input values needed to complete an execution of the software under test

یا grant truth
خروجی حقیقی

مقادیر ورودی مورد نیاز برای تکمیل اجرای نرم افزار تحت آزمایش

- Expected results

نتیجه ای که در صورتی که نرم افزار مطابق انتظار عمل کند، توسط تست حاصل می شود

The result that will be produced by the test if the software behaves as expected

A test oracle uses expected results to decide whether a test passed or failed.

a test oracle uses expected results to decide whether a test passed or failed.

Test case value

چیزی که برای اجرای اون نرم افزار قابل اجرا، موردنیاز است

- Are inputs to the program that test designers use to directly satisfy the test requirements.
- They determine the quality of the testing.
- Definition of test case values is quite broad.
- Test case values are not enough. In addition to test case values, other inputs are often needed to run a test.

ورودی هایی برای برنامه هستند که طراحان آزمون برای برآورده کردن مستقیم نیازهای آزمون از آنها استفاده می کنند.
کیفیت تست را تعیین می کنند.
تعریف مقادیر Test case کاملاً گسترده است.
مقادیر مورد آزمایش کافی نیست. علاوه بر مقادیر مورد آزمایش، ورودی های دیگری نیز برای اجرای یک آزمون مورد نیاز است.

Affecting Controllability and Observability

اینپوت ها را کنترل میکنند

- Prefix values

ورودی های لازم برای قرار دادن نرم افزار در وضعیت مناسب برای دریافت مقادیر مورد آزمایش

Inputs necessary to put the software into the appropriate state to receive the test case values

- Postfix values

روی observablillity تاثیر میگذاره

مثلا برای رزرو کتاب در یک سایت اول باید یوزرنیم و پسورد را بزنیم و باید اول عضو شده باشیم تا بتوانیم رزرو کنیم.

Any inputs that need to be sent to the software after the test case values are sent

چه اینپوت هایی را نیاز داریم تا به استیت ای برسیم که بتوانیم نتایج را مشاهده کنیم.

مقادیری که نیازه به نرم افزار بدیم بعد از اینکه اون تست کیس ولیو ها را به نرم افزار دادیم.

1. *Verification Values* : Values needed to see the results of the test case values

هر ورودی که باید پس از ارسال مقادیر مورد آزمایش به نرم افزار ارسال شود

2. *Exit Values* : Values or commands needed to terminate the program or otherwise return it to a stable state

Verification Values: مقادیر مورد نیاز برای مشاهده نتایج مقادیر مورد آزمایشی

Exit Values: مقادیر یا دستورات مورد نیاز برای خاتمه دادن به برنامه یا بازگرداندن آن به حالت پایدار

Putting Tests Together

- Test case

The test case values, prefix values, postfix values, and expected results necessary for a complete execution and evaluation of the software under test

مقادیر مورد آزمایش، مقادیر پیشوند، مقادیر پسوند و نتایج مورد انتظار لازم برای اجرا و ارزیابی کامل نرم افزار تحت آزمایش

- Test set

A set of test cases

- Executable test script

A test case that is prepared in a form to be executed automatically on the test software and produce a report

یک تست کیس که در فرمی آماده می شود تا به صورت خودکار روی نرم افزار تست اجرا شود و گزارش تهیه شود

Example

- Consider the function *estimateShipping*() that estimates shipping charges for preferred customers in an automated shopping cart application.

- Suppose we are writing tests to check whether the estimated shipping charges match the actual shipping charges.

Example (Cnt'd)

- Prefix values, designed to reach (*R*) the estimateShipping() function in an appropriate state.
  - Involve creating a shopping cart, adding various items to it, and obtaining a preferred customer object with an appropriate address.

مقادیر پیشوند، طراحی شده برای رسیدن به (R) عملکرد برآوردShipping() در یک حالت مناسب.
- شامل ایجاد یک سبد خرید، افزودن اقلام مختلف به آن و به دست آوردن یک شی مشتری ترجیحی با آدرس مناسب است.

- Test case values, designed to achieve infection (*I*), might be the type of shipping desired: overnight vs. regular.

مقادیر مورد آزمایش، طراحی شده برای رسیدن به عفونت! (Infection)،
ممکن است نوع حمل و نقل مورد نظر باشد: یک شبه در مقابل معمولی
خود فانکشن اصلی چک میشه!

منتقل کردن نتیجه و خروجی تابع که از مرحله قبل اومده

ample (Cnt'd)

- Postfix values, designed to achieve propagation (*P*) and make an infection result in an observable failure,
  - might involve completing the order, so that actual shipping charges are computed.
- the revealing part (*R*) of the final order is probably implemented by extracting the actual shipping charge, although there are many other parts of the final order that could also be incorrect.

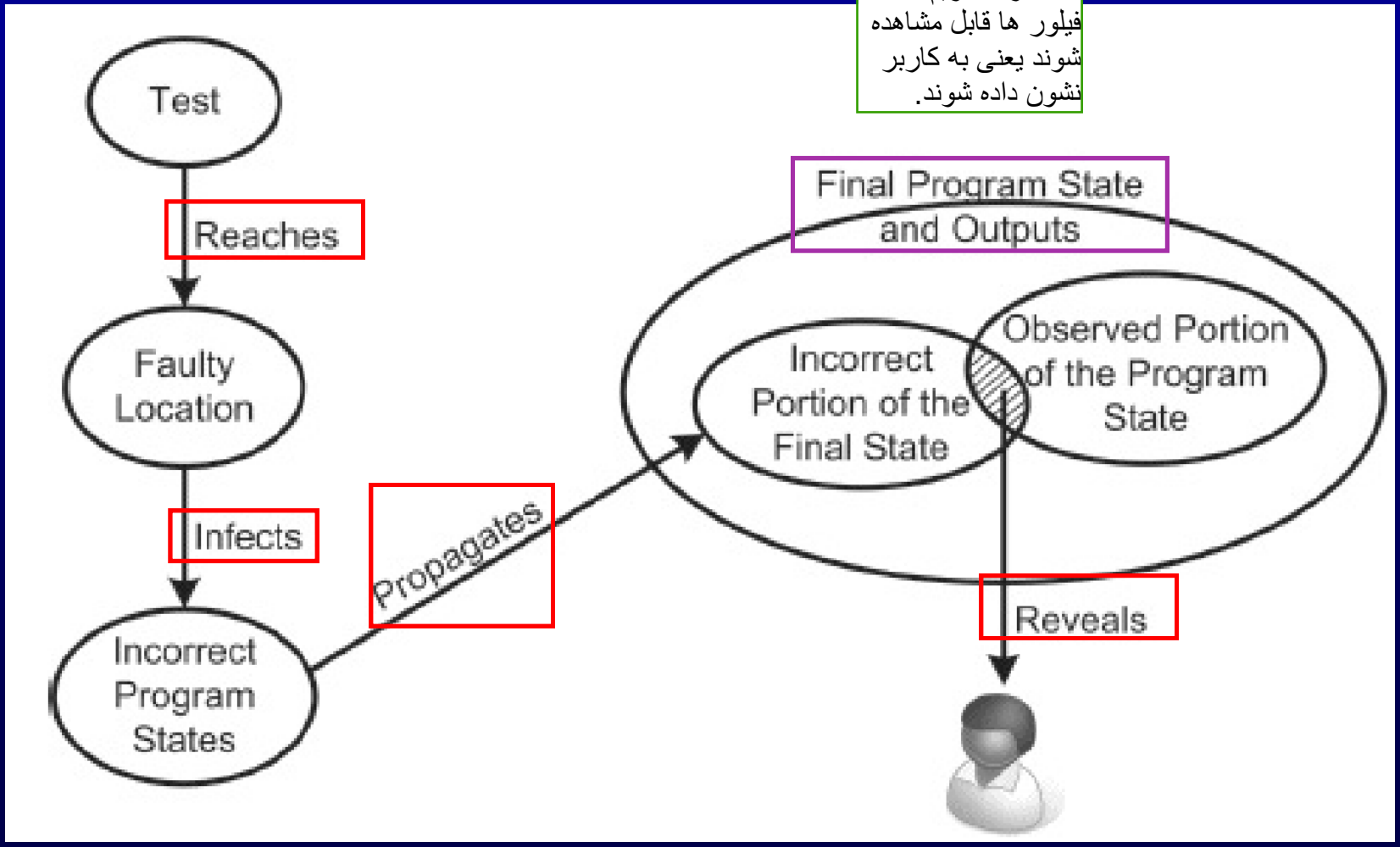
مقادیر Postfix، طراحی شده برای رسیدن به انتشار (P) و ایجاد عفونت منجر به شکست قابل مشاهده می شود.
- ممکن است شامل تکمیل سفارش باشد، به طوری که هزینه های حمل و نقل واقعی محاسبه شود.

قسمت آشکار کننده (R) سفارش نهایی احتمالاً با استخراج هزینه حمل و نقل واقعی انجام می شود، اگرچه بسیاری از بخش های دیگر سفارش نهایی نیز می توانند نادرست باشند.

Test case and RIPR model

- The components in a test case are concrete realizations of the RIPR model.

۴تا شرط داریم تا فیلور ها قابل مشاهده شوند یعنی به کاربر نشون داده شوند.



Test case and RIPR model

A test can be thought of as being designed to look for a fault in a particular location in the program.

یک تست را می توان به عنوان طراحی شده برای جستجوی عیب در یک مکان خاص در برنامه در نظر گرفت.

- Prefix values are included to achieve reachability (*R*),
- Test case values to achieve infection (*I*),
- Postfix values to achieve propagation (*P*),
- Expected results to reveal the failures (*R*).
  - Usually cannot include values for the entire output state of the program, so a well designed test case should check the portion of the output state that is relevant to the input values and the purpose of the test.

معمولاً نمی تواند مقادیری را برای کل وضعیت خروجی برنامه در بر بگیرد، بنابراین یک مورد آزمایشی که به خوبی طراحی شده است باید بخشی از وضعیت خروجی را که با مقادیر ورودی و هدف آزمایش مرتبط است بررسی کند.

Ideally, the automation should be complete in the sense of running the software with the test case values, getting the results, comparing the results with the expected results, and preparing a clear report for the test engineer.

در حالت ایدهآل، اتوماسیون باید کامل باشد به این معنا که نرمافزار را با مقادیر مورد آزمایش، دریافت نتایج، مقایسه نتایج با نتایج مورد انتظار و تهیه گزارش شفاف برای مهندس آزمون، کامل باشد.

- The only time a test engineer would not want to automate is if the cost of automation outweighs the benefits.
  - Happen if we are sure the test will only be used once.
  - If the automation requires knowledge or skills that the test engineer does not have.

تنها زمانی که یک مهندس آزمایشی مایل به خودکارسازی نیست این است که هزینه اتوماسیون بیشتر از مزایای آن باشد.
- در صورتی اتفاق میافتد که مطمئن باشیم از تست فقط یک بار استفاده میشود.
– اگر اتوماسیون به دانش یا مهارت هایی نیاز دارد که مهندس آزمون آن را ندارد.
نمیصرفه که هزینه اموزش را برای اعضای تیم بدهیم تا اون روش خاص اتوماسیون را یادبگیرند و به کار بگیرن