Introduction to Software Testing
(*2nd edition*)
Chapter 1

Why Do We Test Software?

Paul Ammann & Jeff Offutt

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

Updated September 2015
First version, 28 August 2011

Introduction

- To teach software engineers how to test.

- Is useful for
  - a programmer who needs to unit test her own software,
  - a full-time tester who works mostly from requirements at the user level,
  - a manager in charge of testing or development,
  - any position in between.

Software Faults, Errors & Failures

- Software Fault : A static defect in the software

- Software Error : An incorrect internal state that is the manifestation of some fault

خطای نرم افزار: یک نقص استاتیک در نرم افزار
خطای نرم افزار: یک حالت داخلی نادرست که مظهر نقص است
خرابی نرم افزار: رفتار خارجی و نادرست با توجه به الزامات یا سایر توضیحات رفتار مورد انتظار

- Software Failure : External, incorrect behavior with respect to the requirements or other description of the expected behavior

Faults in software are equivalent to design mistakes in hardware.

Software does not degrade.

نقص در نرم افزار معادل اشتباهات طراحی در سخت افزار است.
نرم افزار تخریب نمی شود.

Fault and Failure Example

- A patient gives a doctor a list of symptoms
  - Failures

یک بیمار لیستی از علائم را به پزشک می دهد

- The doctor tries to diagnose the root cause, the ailment
  - Fault

پزشک سعی می کند علت اصلی، بیماری را تشخیص دهد

- The doctor may look for anomalous internal conditions (high blood pressure, irregular heartbeat, bacteria in the blood stream)
  - Errors

پزشک ممکن است به دنبال شرایط داخلی غیرعادی باشد

Most medical problems result from external attacks (bacteria, viruses) or physical degradation as we age.

Software faults were there at the beginning and do not "appear" when a part wears out.

ایرادات نرم افزاری در ابتدا وجود داشت و زمانی که یک قطعه فرسوده می شود، ظاهر نمی شود

A Concrete Example

Fault: Should start searching at 0, not 1

```
public static int numZero (int [ ] arr)
{  // Effects: If arr is null throw NullPointerException
   // else return the number of occurrences of 0 in arr
   int count = 0;
   for (int i = 1; i < arr.length; i++)
   {
      if (arr [ i ] == 0)
      {
         count++;
      }
   }
   return count;
}
```

Test 1
[2, 7, 0]
Expected: 1
Actual: 1

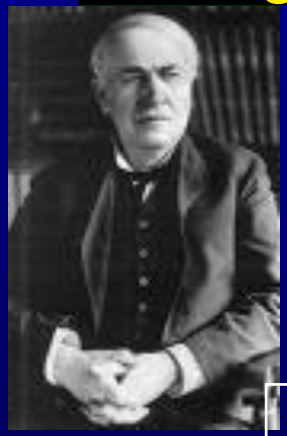
Error: i is 1, not 0, on the first iteration
Failure: none

Test 2
[0, 2, 7]
Expected: 1
Actual: 0

Error: i is 1, not 0
Error propagates to the variable count
Failure: count is 0 at the return statement

The Term Bug

- *Bug* is used informally
- Sometimes speakers mean fault, sometimes error, sometimes failure … often the speaker doesn't know what it means !
- This class will try to use words that have precise, defined, and unambiguous meanings



"It has been just so in all of my inventions. The first step is an intuition, and comes with a burst, then difficulties arise—this thing gives out and *[it is]* then that 'Bugs'—as such little faults and difficulties are called—show themselves and months of intense watching, study and labor are requisite. . ." – Thomas Edison

"an analyzing process must equally have been performed in order to furnish the Analytical Engine with the necessary operative data; and that herein may also lie a possible source of error. Granted that the actual mechanism is unerring in its processes, the cards may give it wrong orders. " – Ada, Countess Lovelace (notes on Babbage's Analytical Engine)

What Does This Mean?

Software testing is getting more important

What are we trying to do when we test ?

What are our goals ?

Validation & Verification (*IEEE*)

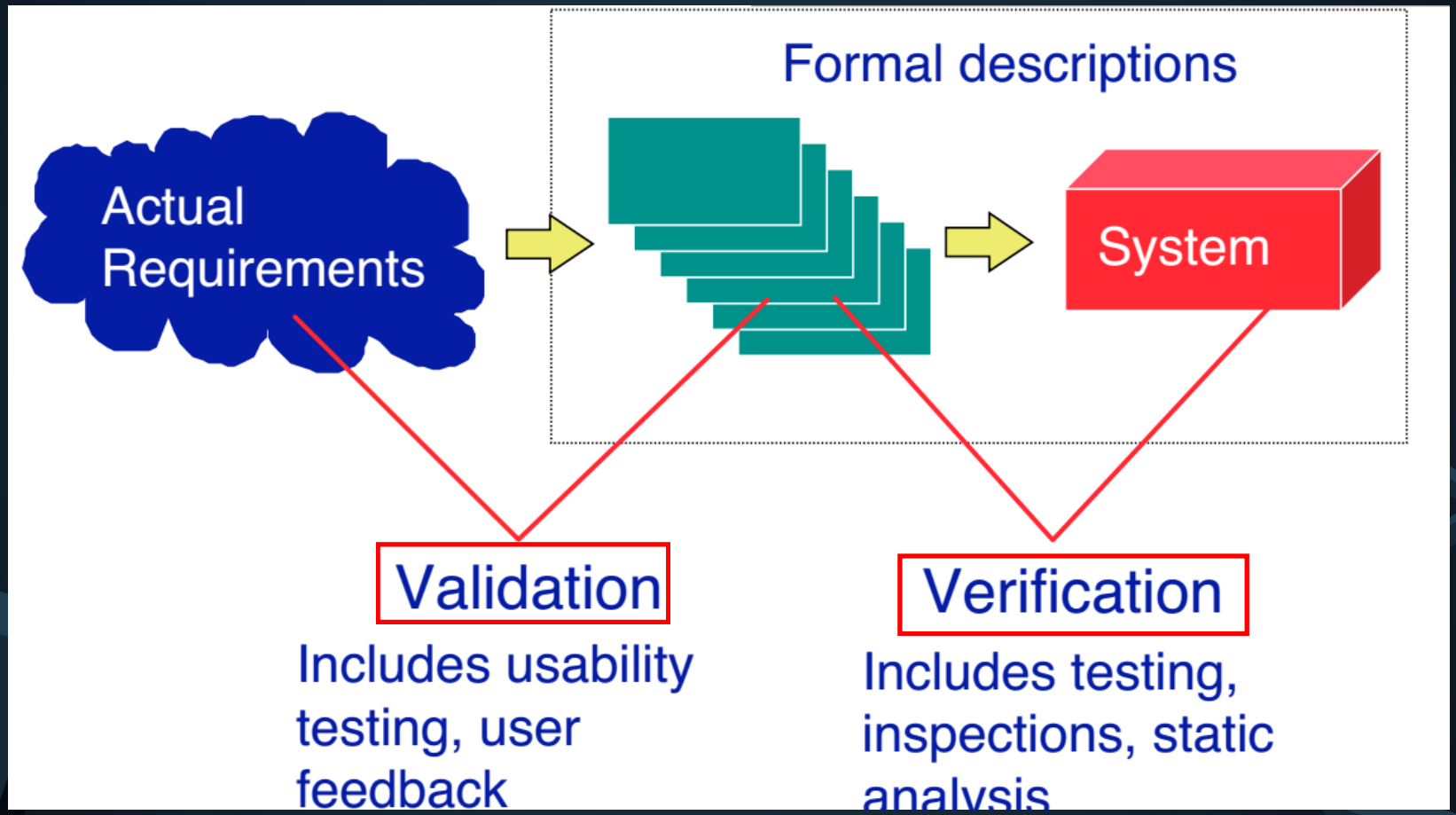
- Validation : The process of evaluating software at the end of software development to ensure compliance with intended usage
  - Depends on domain knowledge; that is, knowledge of the application for which the software is written.

اعتبار سنجی: فرآیند ارزیابی نرم افزار در پایان توسعه نرم افزار برای اطمینان از انطباق با استفاده مورد نظر
- بستگی به دانش حوزه دارد. یعنی دانش برنامه ای که نرم افزار برای آن نوشته شده است.

- Verification : The process of determining whether the products of a given phase of the software development process fulfill the requirements established during the previous phase.
  - Is a more technical activity that uses knowledge about the individual software artifacts, requirements, and specifications.

تأیید: فرآیند تعیین اینکه آیا محصولات یک مرحله معین از فرآیند توسعه نرم افزار الزامات تعیین شده در مرحله قبل را برآورده می کنند یا خیر.
- یک فعالیت فنی تر است که از دانش در مورد مصنوعات، الزامات و مشخصات نرم افزار فردی استفاده می کند.

Validation and Verification



This figure is brought from “Analysis and Testing are Creative”, Dr. Debra Richardson, 2002.

Testing Goals Based on Test Process Maturity

- Level 0 : There's no difference between testing and debugging
- Level 1 :The purpose of testing is to show correctness
- Level 2 :The purpose of testing is to show that the software doesn't work
- Level 3 :The purpose of testing is not to prove anything specific, but to reduce the risk of using the software
- Level 4 :Testing is a mental discipline that helps all IT professionals develop higher quality software

اهداف آزمون بر اساس بلوغ فرآیند آزمون
سطح 0: هیچ تفاوتی بین تست و اشکال زدایی وجود ندارد
سطح 1: هدف از تست نشان دادن درستی است
سطح 2: هدف از آزمایش نشان دادن این است که نرم افزار کار نمی کند
سطح 3: هدف از آزمایش اثبات چیز خاصی نیست، بلکه کاهش خطر استفاده از نرم افزار است.
سطح 4: تست یک رشته ذهنی است که به همه متخصصان فناوری اطلاعات کمک می کند نرم افزار با کیفیت بالاتر توسعه دهند.

Level 0 Thinking

- Testing is the same as debugging
- Does not distinguish between incorrect behavior and mistakes in the program
- Does not help develop software that is reliable or safe

سطح 0 تفکر
تست همان اشکال زدایی است
بین رفتار نادرست و اشتباه در برنامه تمایز قائل نمی شود
به توسعه نرم افزار قابل اعتماد یا ایمن کمک نمی کند
این چیزی است که ما در مقطع کارشناسی رشته های CS آموزش می دهیم

This is what we teach undergraduate CS majors

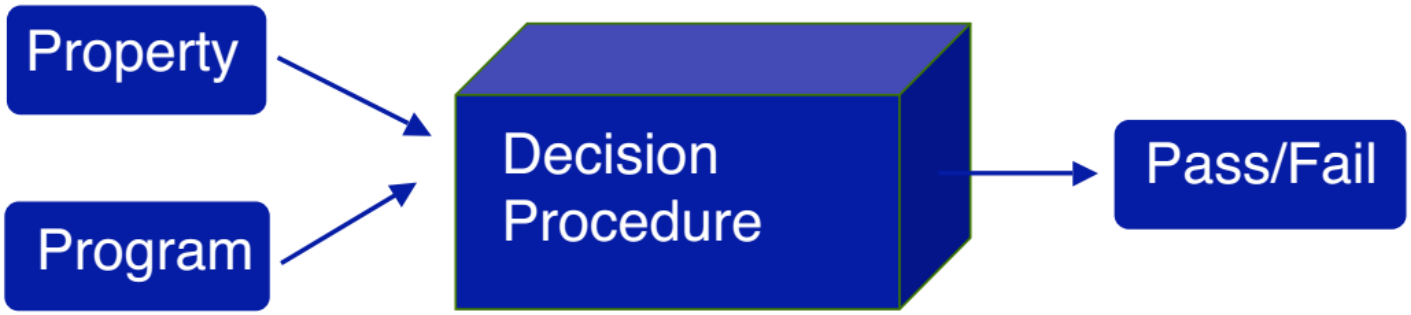
Level 1 Thinking

- Purpose is to show correctness
- Correctness is impossible to achieve
- What do we know if no failures?
  - Good software or bad tests?
- Test engineers have no:
  - Strict goal
  - Real stopping rule
  - Formal test technique
  - Test managers are powerless

سطح 1 تفکر
هدف نشان دادن درستی است
رسیدن به صحت غیرممکن است
اگر شکستی نداشته باشیم چه می دانیم؟
– نرم افزار خوب یا تست بد؟
مهندسین آزمون هیچ:
- هدف دقیق
- قانون توقف واقعی
- تکنیک تست رسمی
- مدیران آزمون ناتوان هستند
این همان چیزی است که مهندسان سخت افزار اغلب انتظار دارند

This is what hardware engineers often expect

You can't ~~always~~ *ever* get what you want



Correctness properties are undecidable

the halting problem can be embedded in almost every property of interest

This figure is brought from "Analysis and Testing are Creative", Dr. Debra Richardson, 2002.

Level 2 Thinking

- Purpose is to show failures
- Looking for failures is a negative activity
- Puts testers and developers into an adversarial relationship
- What if there are no failures?

Level 3 Thinking

- Testing can only show the presence of failures
- Whenever we use software, we incur some risk
- Risk may be small and consequences unimportant

- Risk may be great and consequences catastrophic
- Testers and developers cooperate to reduce risk

سطح 3 تفکر
آزمایش فقط می تواند وجود نقص را نشان دهد
هر زمان که از نرم افزار استفاده می کنیم، ریسک هایی را متحمل می شویم
خطر ممکن است کوچک و عواقب آن بی اهمیت باشد
خطر ممکن است بزرگ و عواقب فاجعه بار باشد
آزمایش کنندگان و توسعه دهندگان برای کاهش ریسک همکاری می کنند
این چند شرکت نرم افزاری "روشنفکر" را توصیف می کند

This describes a few "enlightened" software companies

Level 4 Thinking

A mental discipline that increases quality

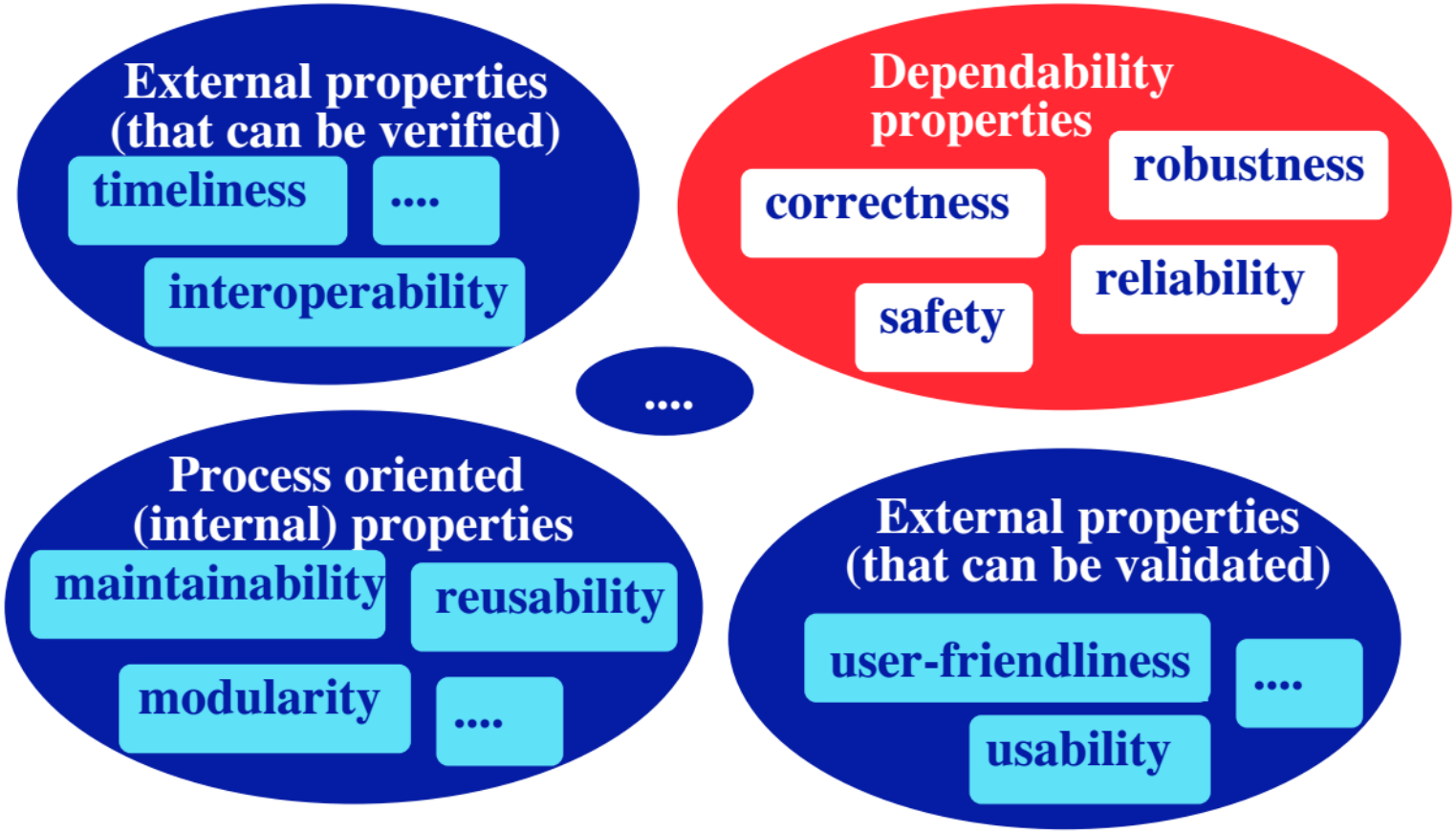
- Testing is only one way to increase quality
- Test engineers can become technical leaders of the project
- Primary responsibility to measure and improve software quality
- Their expertise should help the developers

This is the way "traditional" engineering works

مسئولیت اصلی اندازه گیری و بهبود کیفیت نرم افزار
تخصص آنها باید به توسعه دهندگان کمک کنداین روشی است که مهندسی "سنتی" کار می کند

سطح 4 تفکر
یک نظم ذهنی که کیفیت را افزایش می دهد
تست تنها یک راه برای افزایش کیفیت است
مهندسان آزمون می توانند رهبران فنی پروژه شوند

Software Qualities



This figure is brought from “Analysis and Testing are Creative”, Dr. Debra Richardson, 2002.

Where Are You?

Are you at level 0, 1, or 2 ?

Is your organization at work at level 0, 1, or 2 ?

Or 3?

ما امیدواریم که به شما آموزش دهیم که در محل کار خود "عامل تغییر" شوید ... حامیان تفکر سطح 4

We hope to teach you to become "change agents" in your workplace …

Advocates for level 4 thinking

Tactical Goals : Why Each Test ?

If you don't know why you're conducting each test, it won't be very helpful

- Written test objectives and requirements must be documented

اهداف و الزامات آزمون کتبی باید مستند باشد سطوح پوشش برنامه ریزی شده شما چیست؟

- What are your planned coverage levels?
- How much testing is enough?

اهداف تاکتیکی: چرا هر آزمون؟ اگر نمی دانید چرا هر آزمون را انجام می دهید، خیلی مفید نخواهد بود

- Common objective – spend the budget . . . test until the ship-date …
  - Sometimes called the "date criterion"

چقدر آزمایش کافی است؟
هدف مشترک - صرف بودجه ... آزمایش تا تاریخ ارسال ...
- گاهی اوقات "معیار تاریخ" نامیده می شود

Why Each Test ?

If you don't start planning for each test when the functional requirements are formed, you'll never know why you're conducting the test

- 1980: "The software shall be easily maintainable"

1980: "نرم افزار باید به راحتی قابل نگهداری باشد
"الزامات قابلیت اطمینان آستانه؟
هر آزمون سعی می کند چه واقعیتی را تأیید کند؟

- Threshold reliability requirements?

تیم های تعریف نیازمندی ها به آزمایش کننده نیاز دارند!اگر زمانی که الزامات عملکردی شکل گرفت، برای هر آزمون برنامه ریزی نکنید، هرگز نمی دانید چرا آزمون را انجام می دهید.

- What fact does each test try to verify?

- Requirements definition teams need testers!

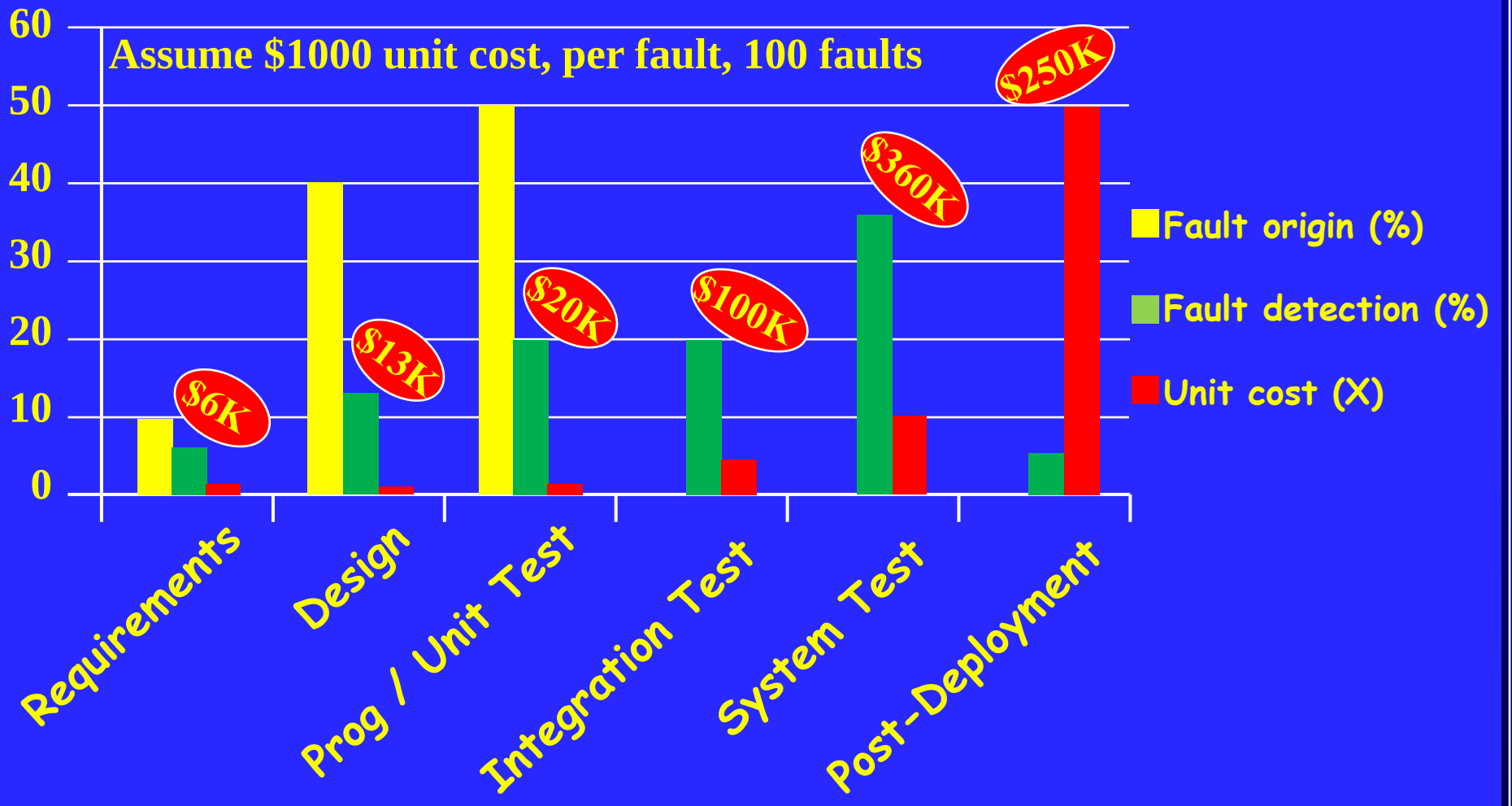
Cost of Not Testing

Poor Program Managers might say:
"Testing is too expensive."

- Testing is the most time consuming and expensive part of software development
- Not testing is even more expensive
- If we have too little testing effort early, the cost of testing increases
- Planning for testing after development is prohibitively expensive

هزینه عدم آزمایش
تست زمان برترین و پرهزینه ترین بخش توسعه نرم افزار است
تست نکردن حتی گرانتر است
اگر تلاش آزمایشی خیلی کمی داشته باشیم، هزینه آزمایش افزایش می یابد
برنامه ریزی برای آزمایش پس از توسعه بسیار گران است
مدیران برنامه ضعیف ممکن است بگویند: "تست بسیار گران است."

Cost of Late Testing



Software Engineering Institute; Carnegie Mellon University; Handbook CMU/SEI-96-HB-002

So...

Avoiding the work early
(requirements analysis and unit testing)
saves money in the short term.

But it leaves faults in software that are like little bombs, ticking away, and the longer they tick, the bigger the explosion when they finally go off.

بنابراین...
اجتناب از کار زود هنگام (تجزیه و تحلیل نیازمندی ها و آزمایش واحد) در کوتاه مدت باعث صرفه جویی در هزینه می شود.
ما در نرمافزارها که مانند بمبهای کوچک هستند، ایرادهایی به جا میگذارد و هر چه بیشتر تیکک میزنند، وقتی در نهایت خاموش میشوند، انفجار بزرگتر میشود.

Impact of Software on Testing

The type of software and its characteristics impact in different ways the testing and analysis activities.

– different emphasis may be given to the same properties
– different (new) properties may be required
– different (new) testing and analysis techniques may be needed

تاثیر نرم افزار بر تست
نوع نرم افزار و ویژگی های آن به طرق مختلف بر فعالیت های تست و تحلیل تاثیر می گذارد.
- ممکن است بر ویژگی های یکسان تأکید متفاوتی داده شود
- ممکن است ویژگی های متفاوت (جدید) مورد نیاز باشد
- ممکن است به تکنیک های مختلف (جدید) آزمایش و تجزیه و تحلیل نیاز باشد

This figure is brought from “Analysis and Testing are Creative”, Dr. Debra Richardson, 2002.

Different emphasis on different properties

الزامات قابلیت اطمینان
- آنها به شدت با هم تفاوت دارند
- برنامه های کاربردی حیاتی ایمنی
- سیستم های کنترل پرواز الزامات ایمنی سختگیرانه ای دارند
- سیستم های مخابراتی الزامات استحکام سختی دارند
- محصولات بازار انبوه
- قابلیت اطمینان کمتر از زمان عرضه به بازار اهمیت دارد
- در یک کلاس از محصولات متفاوت است
- قابلیت اطمینان و استحکام مسائل کلیدی برای سیستم عامل های چند کاربره است و برای سیستم عامل های تک کاربر اهمیت کمتری دارد.

Dependability requirements

- They differ radically between
  - Safety-critical applications
    - Flight control systems have strict safety requirements
    - Telecommunication systems have strict robustness requirements
  - Mass-market products
    - Dependability is less important than time to market
- Vary within the same class of products

• Reliability and robustness are key issues for multi-user operating systems and less important for single users operating systems.

This slide is brought from “Analysis and Testing are Creative”, Dr. Debra Richardson, 2002.

Different type of software may require different properties

- Timing properties
  - deadline satisfaction is a key issue for real time systems,
  - performance is important for many applications
- Synchronization properties
  - absence of deadlock is important for concurrent or distributed systems,
- External properties
  - user friendliness is an issue for GUI, irrelevant for embedded controllers.

- خواص زمان بندی
- رضایت از ضرب الاجل یک مسئله کلیدی برای سیستم های زمان واقعی است،
- عملکرد برای بسیاری از برنامه ها مهم است
- ویژگی های همگام سازی
- عدم وجود بن بست برای سیستم های همزمان یا توزیع شده مهم است،
- خواص خارجی
- کاربر پسند بودن یک مشکل برای رابط کاربری گرافیکی است که برای کنترلرهای تعبیه شده بی ربط است.

This slide is brought from “Analysis and

Different Testing for checking the same properties for different software

- Test selection criteria based on structural coverage are different for
  - procedural software (statement, branch, path,…)
  - object oriented software (coverage of combination of polymorphic calls and dynamic bindings,…)
  - concurrent software (coverage of concurrent execution
  - sequences,…)
- Absence of deadlock can be statically checked on some systems, require the construction of the reachability space for other systems.

- معیارهای انتخاب آزمون بر اساس پوشش سازه متفاوت است
- نرم افزار رویه ای (گزاره، شاخه، مسیر،...)
- نرم افزار شی گرا (پوشش ترکیبی از فراخوانی های چند شکلی و پیوندهای پویا،...)
- نرم افزار همزمان (پوشش اجرای همزمان
- دنباله ها،...)
- عدم وجود بن بست را می توان به صورت ایستا در برخی سیستم ها بررسی کرد، نیاز به ساخت فضای دسترسی برای سایر سیستم ها دارد.

This slide is brought from “Analysis and Testing are

Principles of effective software testing techniques

اصول تکنیک های تست نرم افزار موثر
• حساسیت: بهتر است هر بار شکست بخورید تا گاهی اوقات

- Sensitivity: better to fail every time than sometimes
  - Run time deadlock analysis works better if it is machine independent.

• تحلیل بن بست زمان اجرا اگر مستقل از ماشین باشد بهتر عمل می کند.

- Redundancy: making intentions explicit
  - Increase the capabilities of catching specific faults early or more efficiently

• افزونگی: واضح ساختن مقاصد
• قابلیت تشخیص زودهنگام یا کارآمدتر عیب های خاص را افزایش دهید

- Partitioning: divide and conquer
  - Can be handled by suitably partitioning the input space
- Restriction: making the problem easier
  - Can reduce hard (unsolvable) problems to simpler (solvable) problems
- Feedback: tuning the development process
  - Learning from experience

• تقسیم بندی: تفرقه بینداز و غلبه کن
• می توان با پارتیشن بندی مناسب فضای ورودی کار کرد
• محدودیت: آسان کردن مشکل
• می تواند مسائل سخت (غیر قابل حل) را به مسائل ساده (قابل حل) کاهش دهد
• بازخورد: تنظیم فرآیند توسعه
• یادگیری از تجربه

This slide is brought from “Analysis and Testing are Creative”, Dr. Debra F

Fundamental Principles of software testing(I)

اصول اساسی تست نرم افزار

- Testing show the presence of defects.
  - Regardless of how thoroughly a product or application is tested, no one can guarantee that it is defect-free.
- Exhaustive testing is impossible.
  - It is impossible to test all possible combinations of data, inputs, and test scenarios.
- Early testing
  - Should be as soon as possible
- Defect clustering
  - Most defects are found in few modules.

- آزمایش وجود نقص را نشان می دهد.
- صرف نظر از اینکه یک محصول یا برنامه به طور کامل آزمایش شده است، هیچ کس نمی تواند تضمین کند که بدون نقص است.
- آزمایش جامع غیرممکن است.
- آزمایش تمام ترکیبات ممکن از داده ها، ورودی ها و سناریوهای آزمایش غیرممکن است.
- تست اولیه
- باید در اسرع وقت
- خوشه بندی نقص
- بیشتر عیوب در چند ماژول یافت می شود.

Fundamental Principles of software testing(II)

- Pesticide Paradox
  - Frequently review and update the test cases in order to cover various sections of the projects.
- Testing is dependent on the problem.
  - Same techniques cannot be used for multiple projects.
- The lack of the Fallacy of Errors
  - The software product should be tested not only on its technical aspects, but also on the expectations and needs of users.

- پارادوکس آفت کش ها
- موارد آزمایشی را به طور مکرر بررسی و به روز کنید تا بخش های مختلف پروژه ها را پوشش دهید.
- آزمایش به مشکل بستگی دارد.
- تکنیک های مشابه را نمی توان برای چندین پروژه استفاده کرد.
- فقدان مغالطه خطاها
- محصول نرم افزاری باید نه تنها از نظر جنبه های فنی، بلکه بر اساس انتظارات و نیازهای کاربران نیز مورد آزمایش قرار گیرد.

This slide is brought from the study “Foundations of Software Testing" (ISTQB Certification) by Rex Black, Erick Van Veenendaal, and Dorothy Graham

Summary:
Why Do We Test Software ?

A tester's goal is to eliminate faults as early as possible

We can never be perfect, but every time we eliminate a fault during unit testing (or sooner!), we save money.

هدف تستر این است که عیوب را در اسرع وقت از بین ببرد
ما هرگز نمیتوانیم کامل باشیم، اما هر بار که در طول آزمایش واحد (یا زودتر!) یک عیب را برطرف میکنیم، در هزینه صرفهجویی میکنیم.

- بهبود کیفیت
- کاهش هزینه
- حفظ رضایت مشتری

- **Improve quality**
- **Reduce cost**
- **Preserve customer satisfaction**