Chapter 1: Why Do We Test Software?

Authors*: Paul Ammann & Jeff Offutt*

Lecturer*: Shirin Baghoolizadeh*

نمره دهی:
حضور و غیاب
کوییز اجباری داریم
کوییز یهویی داریم --> نمره اضافه میشه --> اگر تا قبل از میان ترم باشه نمره میان ترم رو تکمیل می کنه و نمیشه سیوش کرد واسه پایان ترم

میان ترم: 5 نمره
پایان ترم: 7 نمره
کوییز اجباری: 2 نمره
تکلیف: 2 نمره
پروژه: 4 نمره
کوییزهای یهویی: 1.5 نمره (اضافه بر 20)

Testing in the 21st Century

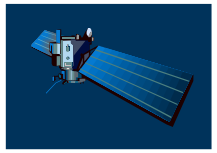
- Software defines behavior
  - network routers, finance, switching networks, other infrastructure
- Today's software market:
  - is much bigger
  - is more competitive
  - has more users
- Embedded Control Applications
  - airplanes, air traffic control
  - spaceships
  - watches
  - ovens
  - remote controllers
- Agile processes put increased pressure on testers
  - Programmers must unit test – with no training or education!
  - Tests are key to functional requirements – but who builds those tests ?

-
نرم افزار همه جا هست و نمیشه ازش فرار کرد مثل ساعت های هوشمند و..
بعضی از نرم افزارها حساس ترند و بعضی هاشون نه --> نرم افزارهایی که خیلی حساس اند با جون ادم سر و کار داره مثل ماشین های خودران..
و بعضی از نرم افزارها هم مهم نیستند مثل پروژه های ترم یک ما
متدلوژی های ایجاد نرم افزار از وقتی که بحث agile پیش اومد بحث تست حساس تر شد
متدلوژی نرم افزار: متدلوژی ینی فرایند و روش --> برای ایجاد نرم افزار رویکردهای مختلفی وجود داره --> رویکردهای نرم افزار: مثلا اول ریکوارمنت ها استخراج میشن و... - رویکردهای دیگه مثل اسکرام و..
متدلوژی agile محوریتشون روی اینه که پروژه رو می شکنن به تسک های کوچیک -->بعد تسک ها رو مثلا می گن تا یه هفته قراره این تسک ها رو انجام بدیم و هر کس از اعضای تیم براساس مهارتش میاد یکی از تسک ها رو انتخاب میکنه
توی متدلوژی agile چون همه چیز سریع است ممکنه خرابکاری زیاد پیش بیاد واسه همین به تست خیلی بها داده میشه توی متدلوژی Agile
حالا کی تست رو انجام بده؟ تست رو میشه توی چند لایه انجام داد:
دو تا نقش است که می تونن تست رو انجام بدن: یکی این که خود دولوپر تست هارو انجام بده یا اینکه نه تستر بیاد و تست رو فقط انجام بده --> کدومش بهتره؟ جفتش باید باشه (بهترین روش این است که ترکیبی با هم کار بکنند)
دولوپر ها در سطح unit test میان تست رو انجام میدن مثلا یک فانکشن رو نوشته و خودش تست می کنه اصلا کار میکنه یا نه ولی وقتی که بحث عظیم میشه ممکنه یک تستر بیاد این وسط

Software is a Skin that Surrounds Our Civilization

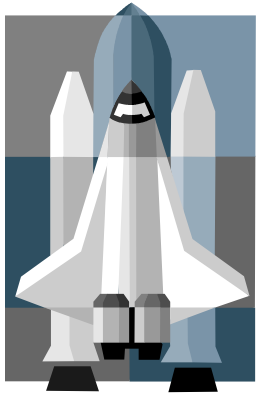
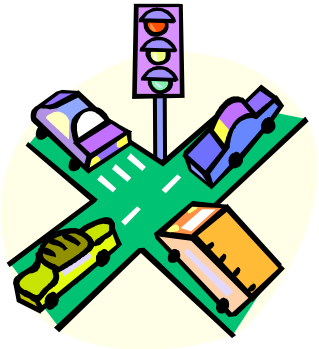










Quote due to Dr. Mark Harman

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

Faults in software are *design mistakes*. They do not appear spontaneously, but exist as a result of a decision by a human.

-
Software Fault : ولی بعضی وقتا درون نرم افزار داریم خرابکاری میکنیم ولی نتیجه درست در میاد --> پس Fault میشه سر منشا تمام اررورها (ممکنه یک ارور توی برنامه توی خروجی در رفتار ظاهری خودش رو نشون بده و ممکنه هم نشون نده)
Fault ینی اشتباهات طراحی که بعد باعث میشه ارور به وجود بیاد
Software Error : استیت داخلی غلط میشه ارور و به سر منشاش می گن fault
Software Failure : ینی توی رفتار ظاهری نرم افزار دیدیم خرابکاری شده

نکته تمرین:
fault اجرا نشه --> ینی اگر این فالت توی خط 13 بود باید یه تست کیسی بگیم که اصلا به این خط نرسه
fault اجرا بشه ولی ارور نده --> ینی وضعیت داخلی کد درست باشه که این وضعیت شامل pc هم می شه مثلا
ارور داشته باشه ولی failure نداشته باشیم --> استیت داخلی غلط بشه ولی توی ظاهر نشون نده غلط رو مثلا توی حلقه i از یک شروع بشه این همیشه میشه ارور ولی ممکنه منجر به failure نشه

Fault and Failure Example

- A patient gives a doctor a list of symptoms
  - Failures
- The doctor tries to diagnose the root cause, the ailment
  - Fault
- The doctor may look for anomalous internal conditions (high blood pressure, irregular heartbeat, bacteria in the blood stream)
  - Errors

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.

-
مثال:
یک نفر مریضه می ره پیش دکتر و دکتر بهش میگه چقدر بی حالی و دکتر از علائم ظاهری اینو گفته که این میشه failure
بعد دکتر ازش ازمایش خون می گیره بعد توی ازمایش خون یکسری چیزا رو چک میکنه و اگر اونایی که چک میکنه مطابق با مقدار استاندارد نباشه این معادل ارور میشه ینی ممکنه یک استیت داخلی در نرم افزار داشته باشیم که این استیت داخلی اون استیت مورد انتظار صحیح نباشه
سر منشا این که مثلا توی اون ازمایش خون یکی از معیارها رفته بالا اینه که مشکل ژنتیکی بوده پس سرمنشا ارور میشه Fault

A Concrete Example

A *program state* is defined during execution of a program as the current value of all live variables and the current location, as given by the program counter.

```
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;
}
```

Fault: Should start searching at 0, not 1

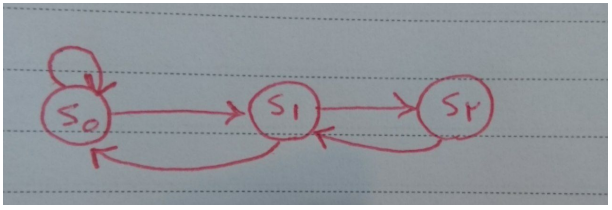
Test 1
[2, 7, 0]
Expected: 1
Actual: 1

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

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

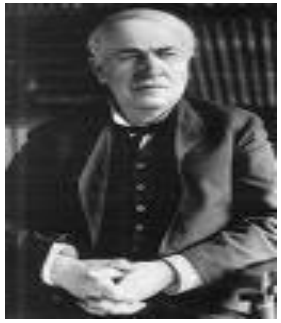
Error: i is 1, not 0
Error propagates to the variable count
Failure: count is 0 at the return statement

-
این برنامه می خواد چک بکنه توی یک ارایه چندتا عنصر صفر وجود داره
سرمنشا همه ارورهاش اینه که برنامه نویس وقتی که میخواسته بچرخه روی ارایه اندیس شروع ارایه رو از یک گذاشته شده ینی عنصر خونه صفر رو در نظر نگرفته
پس Faultاش همینه که i از یک شروع شده
حالا ما برنامه رو تست کردیم و هرچی ارایه بهش میدیم درست جواب میده بخاطر اینکه اولیش صفر نبوده ینی اینکه Failure رخ نداده ینی توی خروجی این fault ظاهر نشده
حالا اگر ارایه بعدی عنصر اولش صفر باشه Failure رخ میده ینی fault اینجا باعث Failure شده
یک سیستم بانک رو می خوایم با گراف مدل سازی بکنیم وضعیت هایی که برای متصدی بانک می تونیم در نظر بگیریم اینه که یا بیکاره یا مشتری داره --> یک وقت استیت سیستم اینه که متصدی بانک مشتری نداره و یک وقت مشتری داره و استیتش عوض میشه --> شکلش پایین صفحه
احتمال اینکه دوتا مشتری همزمان بیاد طبق توزیع پواسون صفر است
استیت های سیستم نرم افزار: ینی توی هر استیت مقدار متغییرهاش چی باشه و مقدار program counter ینی مجموع متغییرهایی که زنده هستند و program counter، استیت یک سیستم نرم افزاری رو برای ما مشخص میکنه
استیت اول مورد انتظار چیه؟ اگر قرار باشه نرم افزار روبه رو مدل بکنیم با گراف ها استیت اولش اینه که توی رفتار صحیح i باید برابر باشه با صفر توی استیت اول چون توی هر تستی i برابر با صفر نیست توی حالت اولیه میگیم توی این برنامه کل ارور وجود داره و اگه توی خروجی هم خودش رو نشون داده میشه failure



The Term Bug

- *Bug* is used informally
- Bug Sometimes means fault, sometimes error, and sometimes failure … often the speaker doesn't know what it means !



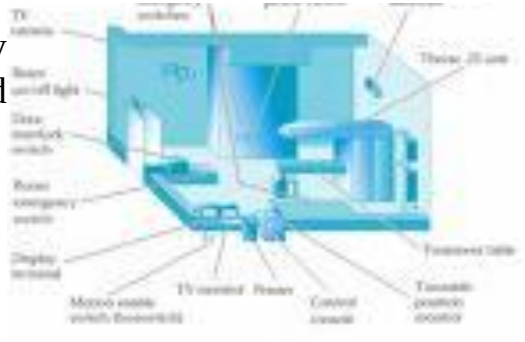
"It has been just so in all of my inventions. The first step is an intuition, and comes with a burst, then difficulties arise—this thing gives out and *[it is]* then that 'Bugs'—as such little faults and difficulties are called—show themselves and months of intense watching, study and labor are requisite.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

---

Spectacular Software Failures

- NASA's Mars lander: September 1999, crashed due to a units integration fault
  - misunderstanding in the units of measure used by two modules created by separate software groups. One module computed thruster data in English units and forwarded the data to a module that expected data in metric units.

THERAC-25 design



- THERAC-25 radiation machine : Poor testing of safety-critical software can cost *lives* : 3 patients were killed
  - due to excessive radiation

- Ariane 5 explosion : About 370 million $ lost
  - exploded 37 seconds after liftoff in 1996. The low-level cause was an unhandled floating point conversion exception in an inertial guidance system function. It turned out that the guidance system could never encounter the unhandled exception when used on the Ariane 4 rocket

- Intel's Pentium FDIV fault : Public relations nightmare

We need our software to be dependable
Testing is *one* way to assess dependability

-

NASA’s Mars lander یا کاوشگر مریخ ناسا: مشکلش این بوده که دوتا ماژول که داشتن کار می کردن این دوتا ماژول توسط دوتا گروه نرم افزاری متفاوت ایجاد شده بود و این دوتا ماژول با هم هماهنگ نبودن و اتفاقی که افتاد این بوده که دیتایی که از ماژول اول باید محاسبه می شده و پاس داده میشده به ماژول دوم چون هر ماژول استانداردش فرق داشته مثلا ماژول اول اینچ بوده اون یه چیز دیگه ینی محاسبات توی هر ماژول درست بوده ولی تعامل دوتا ماژول چون یکی نبوده باعث شده کرش بکنه

THERAC-25 radiation machine یا دستگاه تشعشع THERAC-25: دستگاه پرتو درمانی و باعث شده 3 تا از مریض ها جونشون رو از دست بدن چون نرم افزار محاسباتش اشتباه بوده و باعث میشده که پرتوها بیش از حد به بیمار برسن

Ariane 5 explosior یا انفجار آریان 5: راکت اریان 5 که 370 میلیون دلار باعث خسارت شده --> ورژن قبلی نرم افزار رو داشتن ینی اریان 4 و گفتن اینو توسعه میدیم و یکسری قابلیت ها بهش اضافه میکنیم و تبدیلش میکنیم به اریان 5 و دیگه برای اریان 5 تستش نکردن چون روی اریان 4 به خوبی کار میکرده و گفتن روی اون اوکی بوده دیگه واسه 5 تستش نکردن و این باعث شد 37 ثانیه بعد از پرتاب راکت منفجر بشه

Intel’s Pentium FDIV fault: بعد از ساخت میکروپردازنده متوجه شدن توی هر یک 9 هزار میلیارد خطا داره توی محاسبات

نکته: نرم افزار ما لازمه که قابل اعتماد باشه --> تست کردن یکی از راه های محاسبه قابل اعتماد بودن نرم افزار هستش

Northeast Blackout of 2003

508 generating units and 256 power plants shut down

Affected 10 million people in Ontario, Canada

Affected 40 million people in 8 US states

Financial losses of $6 Billion USD

The alarm system in the energy management system failed due to a software error and operators were not informed of the power overload in the system

توی سال 2003 یک خاموشی بزرگ اتفاق افتاد
چرا خاموشی؟ بخاطر خطای نرم افرار و خطای انسانی --> درخت هایی که رشد کردن خوردن به کابل های برق و از کار افتادن و این باعث شد لود روی کابل های دیگه بسیار زیاد بشه و اینجا باید نرم افزار ارور میده به اپراتورها که بهشون بگه لود زیاد شده ولی نرم افزار fault داشته و تشخیص نداده و ارور نداده و این باعث شده خطوط یکی یکی از کار بیوفته و یک خاموشی بزرگ به وجود بیاد

Spectacular software Failures

- Boeing A220 : Engines failed after software update allowed excessive vibrations

- Boeing 737 Max : Crashed due to overly aggressive software flight overrides (MCAS)
- Toyota brakes : Dozens dead, thousands of crashes

- Healthcare website : Crashed repeatedly on launch—never load tested
- Northeast blackout : 50 million people, $6 billion USD lost … alarm system failed

Software testers try to find faults before the faults find users

---

Validation & Verification (*IEEE*)

- Validation : The process of evaluating software at the end of software development to ensure compliance with intended usage

- 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

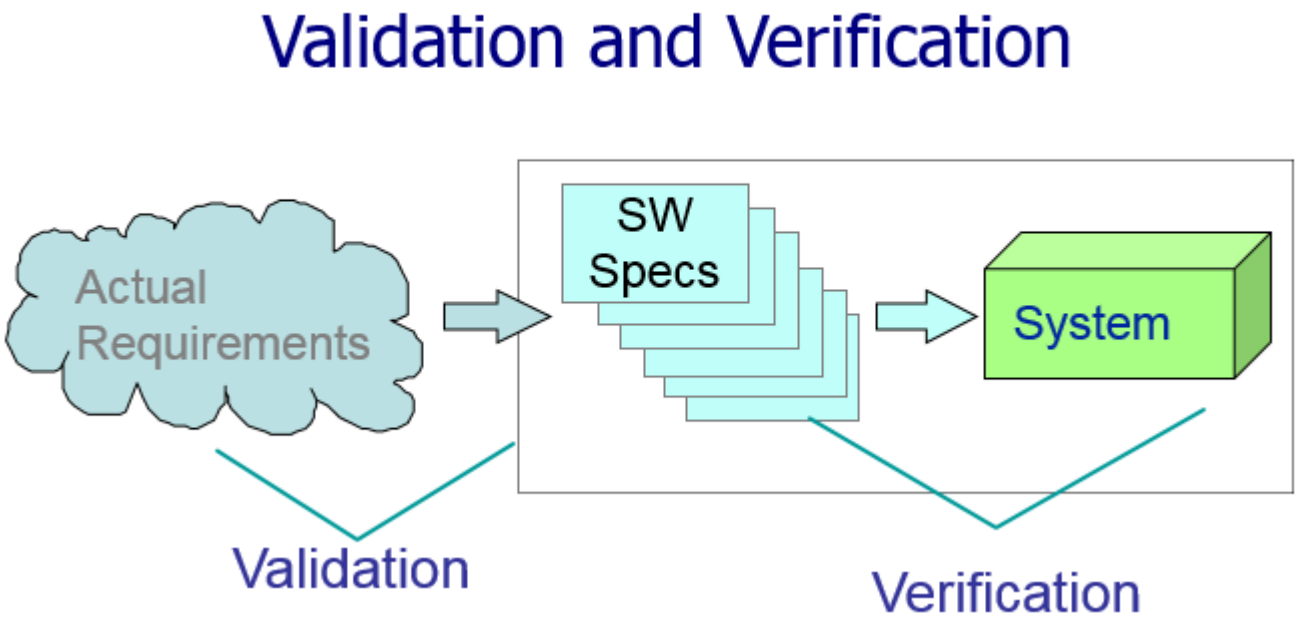
Validation usually depends on domain knowledge; that is, knowledge of the application for which the software is written. For example, validation of software for an airplane requires knowledge from aerospace engineers and pilots.

Verification is usually a more technical activity that uses knowledge about the individual software artifacts, requirements, and specifications.

-
Verification : براساس مدل های ریاضی اثبات میکنه نرم افزار درسته و همونیه که میخوایم

Validation رو وقتی که میخوایم انجام بدیم واقف باشیم بر نیازهای واقعی محیطمون ینی اگه قراره ما نرم افزار هواپیما رو بنویسیم Validation رو کسی انجام بده که با اصول خلبانی اشنا است پس کسی باید Validation رو انجام بده که مسلط باشه به domain knowledge ینی همون نیازهای واقعی محیط با استفاده از دانشی که داره بشناسه

Validation and Verification



فرق Validation و Verification:
Validation : ینی وقتی که ما یک نیازهای واقعی از توی محیط داریم ینی کاربر ما که یکسری نیازها داره و این هارو تبدیلش میکنیم به یک اپلیکیشنی که داره نیاز ها رو جواب میده --> این ینی بریم ببینیم دسته اخر اون نیازهای کارفرما براورده شده یا نه
محصول حتما باید نهایی بشه و بعد Validation بشه
Verification : ینی این که با دوستمون که با دوست خودش بوده چک بکنه ینی توی هر مرحله از پیشرفت نرم افزار با مرحله قبل چک بکنیم که ایا خواسته ها براورده شدن یا نه --> مثلا الان داریم طراحی میکنیم ینی طراحی سطح پایین با جزئیاته و چک بکنیم با طراحی سطح بالا و ببینیم داریم درست می ریم یا نه و طراحی سطح بالا رو چک بکنیم با ریکوارمنت ها و ببینیم داریم درست می ریم یا نه --> Verification میاد توی مراحل داخلی یکی یکی بررسی میکنه
نکته: Verification رو فقط برای نرم افزارهایی که حساس هستن انجام میدیم چون بسیار زمان بر است
نکته: جفتش رو برای نرم افزار می خوایم ولی حضور Verification الزامی نیست ولی Validation رو حتما میخوایم --> Verification همیشه لازم نیست و وقتی که لازم است که میخوایم ریسک استفاده کردن از نرم افزار رو به شدت کاهش بدیم
همه مسیرها رو توی تست نمی تونیم چک بکنیم ولی با Verification می تونیم همه مسیرها رو چک بکنیم
مثال:
دوره امتحانت هست و به دوستمون میگیم میشه برام کیف پولی بخری و دوست هم می ره بیرون ولی یادش می ره و اونم به دوستش زنگ می زنه و میگه میشه برام یک کوله پشتی برای دوستم بخری و با دوستش هم چک میکنه و دوست دوستمون می ره توی فروشگاه و به دوستمون زنگ می زنه و بهش میگه کوله پشتی میخواستی دیگه اره و اونم می گه اره و بعد به نفر اول نشون میدن و میگه من کوله پشتی نمی خواستم کیف پولی می خواستم

IV&V

- IV&V stands for "*independent verification and validation*" where "independent" means that the evaluation is done by non-developers:

  - Sometimes the IV&V team is within the same project,
  - Sometimes the same company,
  - And sometimes it is entirely an external entity.

-
: independent verification and validation
اون کسایی که دارن verification and validation رو انجام میدن مطمئن باشیم که خودشون جزئی از دولوپرها نیستن
کسایی که دولوپر نیستن ممکنه توی همون گروه ما باشن یا ممکنه توی همون شرکت با ما همکار باشن ولی توی گروه ما نباشن یا خارج از شرکت باشن

Testing Goals Based on Test Process Maturity

Beizer discussed the goals of testing in terms of the "test process maturity levels" of an organization, where the levels are characterized by the testers' goals:

- 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

-
تست نرم افزار بستگی داره که چقدر ما عقلمون به بلوغ رسیده باشه:
5 سطح داره:
سطح 0: عقلشون محدودیت داشته
هر چقدر بلوغ فکری پیدا کردیم اهداف تست هم بهتر شدن باعث شده که خطاها زودتر شناسایی بشن و مشکلات نرم افزاری کم بشه

Level 0 Thinking

- Testing is the same as debugging
- This is what we teach undergraduate CS majors
- Does not help develop software that is reliable or safe

-
سطح 0 از بلوغ فکری در تست نرم افزار اینه که یک نفر دیدش اینطوری باشه که تست دارم می کنم و این همون معادل دیباگ کردن کدم است ینی کد رو می نویسیم و بعد می ذاریم دیباگ و بعد یه دونه ران هم ازش میگیریم با اون ورودی هایی که کارفرما داده و جواب هم میگیرم پس تموم شد و تست کردیم دقیقا همون کاری است که توی برنامه نویسی ترم یک انجام می دادیم
نمی تونیم از یک چنین سطح فکری که شخص راجع به تست داریم انتظار داشته باشیم که نرم افزارش یک نرم افزار قابل اعتماد یا ایمن باشه

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: نشون بدیم که برنامه درست است
اینجا هی از خودمون ورودی تولید میکنیم و میدیم به برنامه تا ببینیم نتیجه درست است یا نه
نکته: این خیلی شبیه این است که دنبال خطا بگردیم ولی یک مرز باریکی است که ما دنبال این باشیم که نشون بدیم نرم افزار درست کار میکنه یا نشون بدیم نرم افزار دچار Failure شده مثلا نرم افزار یک عدد x رو میگیره که x می تونه هم منفی باشه و هم صفر باشه و هم مثبت باشه و اگر ما دنبال این باشیم که درست کار بکنه می تونیم x هایی بدیم که فقط مثبت باشه ولی وقتی که می خوایم نشون بدیم Failure داره اون موقع چالش برانگیز میشه واینجا همه بازه ها از اعداد رو بهش میدیم توی نرم افزار مبحث تست اثبات اینکه نرم افزار داره کار میکنه با Verification هستش و با تست اثبات درستی امکان پذیر نیست
فرقش با حالت قبلی این بود که توی سطح 0 تست ها محدود است و به برنامه اش همون ورودی هایی داده که کارفرما بهش داده ولی توی سطح 1 خودش می ره یکسری ورودی هایی علاوه بر اونها می سازه
اینجا تستر نداریم!!

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?

-
سطح 2: نشون بدیم برنامه درست نیست
تستر دنبال اینه که روی نقاط چالش برانگیز دست بذاره تا failure های سیستم رو تشخیص بده
اینجا ممکنه دعوا پیش بیاد بین تستر و برنامه نویس --> برنامه نویس ممکنه فکر کنه که تسترها رقیب هاش هستن و بخوان مچش رو بگیرن ولی سطح بالغ تر فکر اینه که برنامه نویس، تسترها رو به عنوان همکار ببین
اینجا تستر داریم!!

Level 3 Thinking

- Testing can only show the presence, but not the absence, 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:
تستر و دولوپر با هم دوست هستن و می دونن که اون یکی نمیخواد بیاد مچ اون یکی رو بگیره
ما به عنوان یک مهندس نرم افزار می دونیم که اگر Verification انجام ندیم در نهایت فقط متکی باشیم به تست استفاده از نرم افزار یک ریسکی داره و ما این ریسک رو می پذیریم --> حالا توی بعضی از نرم افزارها این ریسک خیلی کوچیک است و عواقب کمی داره ولی توی بعضی هاش بزرگ است
هدف این است که ریسک استفاده از نرم افزار کاهش پیدا کنه

Level 4 Thinking

- A mental discipline that increases quality
- Testing is only one way to increase quality
- Test engineers have primary responsibility to measure and improve software quality
- Their expertise should help the developers

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

Advocates for level 4 thinking

-
سطح 4:
همه می دونن که تست فقط این نیست که ورودی بدیم و خروجی بگیریم بلکه باید نان فانکشنال ریکوارمنت ها رو لحاظ بکنیم و به عنوان یک جریان فکری باشه که از همون ابتدای کار که داریم ریکوارمنت ها رو درمیاریم تسترها هم درگیر باشن چون می خوایم یک نرم افزار باکیفیت داشته باشیم

رفتیم سرکار سطح 4 رو پیش بگیریم...

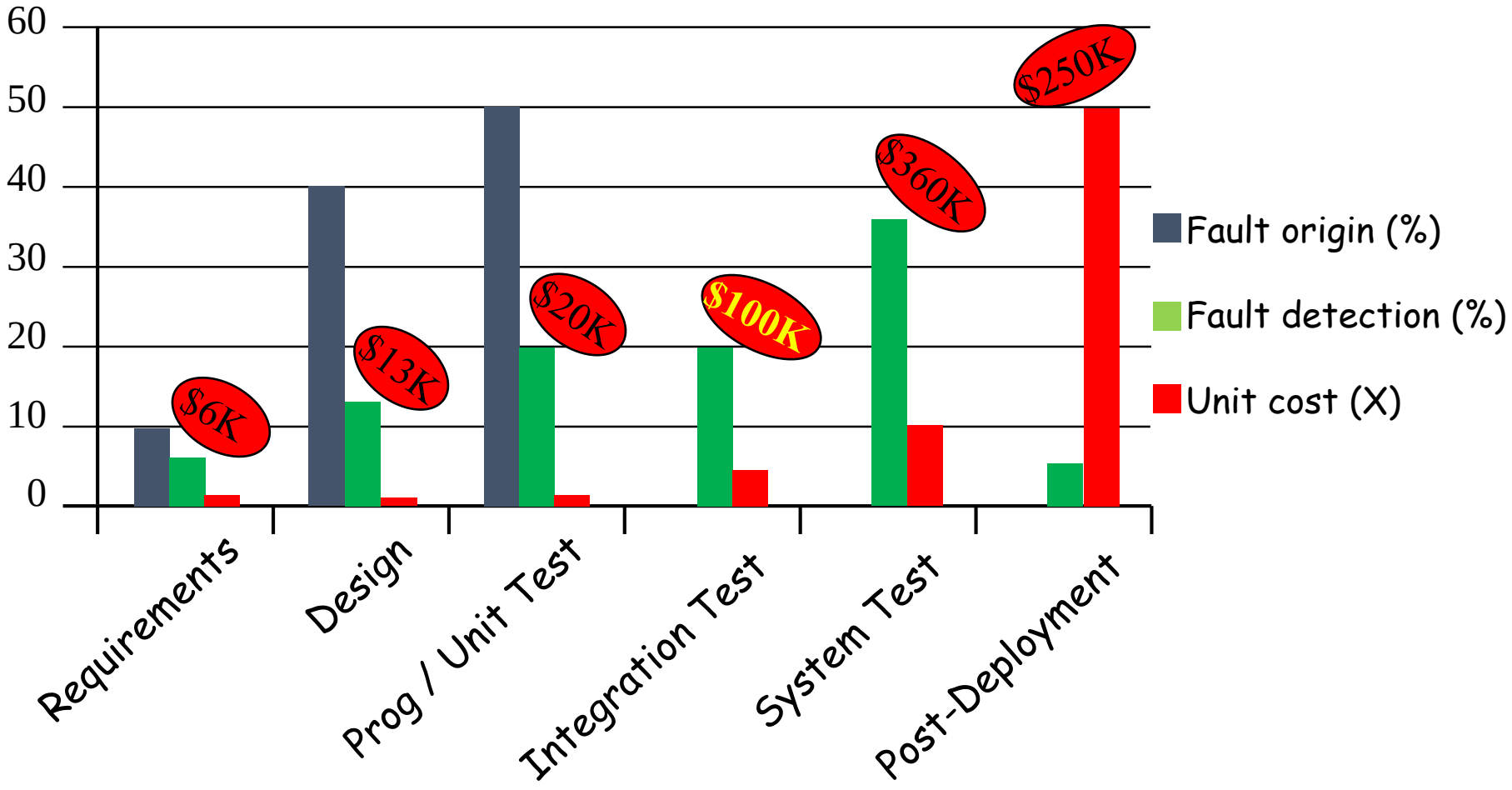
Tactical Goals : Why Each Test ?

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

- Written test objectives and test requirements must be documented
  - What are your planned coverage levels?
  - How much testing is enough?
  - Common objective – spend the budget … test until the ship-date …

-
یکی از مطالبی که اینجا یاد میگیریم اینه که بفهمیم به عنوان کسی که توی گروه تست هستیم باید تست ها هم داکیومنت بشن ینی نه تنها دولوپرها باید کارشون رو داکیومنت بکنن بلکه تسترها هم باید داکیومنت بکنن مثلا بگیم چند درصد از تست برای ما کافی است اینجا یا...

Cost of Late Testing



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

این نمودار خیلی مهم است
از این نظر که ما بفهمیم هر چه زودتر تست رو انجام بدیم هزینه کاهش پیدا میکنه
توی فاز ریکوارمنت می گه توی این فاز چند درصد از خطاها رو به وجود میاره --> 10 درصد از خطاهای کل فرایند بخاطر اشتباهاتی است که توی فاز ریکوارمنت انجام دادیم
40 درصد خطاها مربوط به فاز دیزاین هستن ینی درست دیزاین نکردیم
توی مرحله prog/unit test برای خطا 50 درصد است
و بقیشون خود به خود خطا خیز نیستن
درسته که 10 درصد خطاها توی فاز ریکوارمنت ایجاد میشن ولی اگر ما بریم شناساییشون بکنیم تا یه حدی بیشتر از 50 درصدشون رو همون لحظه می فهمیم ینی کار رو توی همون فاز ریکوارمنت تجزیه و تحلیل بکنیم و بدیم دست تستر و بگیم چک بکن همه چیز رو و چقدر هزینه داریم اینجا یه کوچولو که هزینه unit test است
توی فاز دیزاین درسته که بسیار خطا خیز است ولی خیلی هاش نمی تونیم پیدا بکنیم ولی باز هزینه برای اصلاح و تست کردن کم است

هرچقدر ما زودتر به فکر تست کردن باشیم هزینه ها کاهش پیدا میکنه

Cost of Not Testing

- 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

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

-
بیشترین هزینه رو باید توی فاز تست بدیم

نکته: فرق Errors با Fault:
fault ینی یک خطایی داشته باشیم که هنوز خودشو ظهور نداده مثلا توی شرطمون باید یک علامت بزرگتر می ذاشتیم ولی به اشتباه یک علامت کوچیکتر گذاشتیم --> این سرمنشا ارور و شاید Failures باشه
این fault می تونه باعث ارور بشه یا نشه
ارور: ینی استیت داخلی برنامه با اون استیت مورد انتظار متفاوت باشه