دانشکده مهندسی کامپیوتر

عنوان درس:

طراحی نرم‌افزارهای اتکاءپذیر

(Dependable Software Design)

فصل ۲: مبانی اتکاءپذیری

مدرس: محمد عبداللهی ازگمی

(Mohammad Abdollahi Azgomi)

azgomi@iust.ac.ir

Fundamentals of Dependability

- **Reference:**
  - **E. Dubrova, *Fault-Tolerant Design: An Introduction*, Kluwer Academic Publisher (2005)**
    - Chapter 2: Fundamentals of Dependability
- ----------
- *Ah, this is obviously some strange usage of the word 'safe' that I wasn't previously aware of.*

  —Douglas Adams, "The Hitchhikers Guide to the Galaxy".

Contents

Paper Review Assignment

- **[ALRL]** A. Avizienis, J.-C. Laprie, B. Randell and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing,", *IEEE Trans. on Dependable and Secure Computing* **1**(**1**) (2004) 11-33
  - یکی از مقاله‌های مهم است که جزء مراجع درس محسوب می‌شود.
  - توسط همه دانشجویان مطالعه شود.
  - توسط آقای مومنی ارائه شود.
  - موعد ارائه: ۸۵/۱۱/۱۴
  - مقالات مشابهی متعاقباً انتخاب خواهد شد.

1. Introduction

- The ultimate goal of fault tolerance is the **development of a dependable system**.
  - هدف نهایی تحمل‌پذیری خطا، ساخت سیستم‌های اتکاءپذیر است.
- In a broad term, ***dependability*** is the ability of a system to deliver its intended level of service to its users.
  - اتکاءپذیری عبارت است از توانایی یک سیستم برای ارائه سطح سرویس مورد نظر به کاربرانش.

1. Introduction

- As computer systems become relied upon by society more and more, dependability of these systems becomes a critical issue.
- In airplanes, chemical plants, heart pace-makers (دستگاه‌های تنظیم ضربان قلب) or other safety critical applications, a system failure can cost people's lives or environmental disaster.

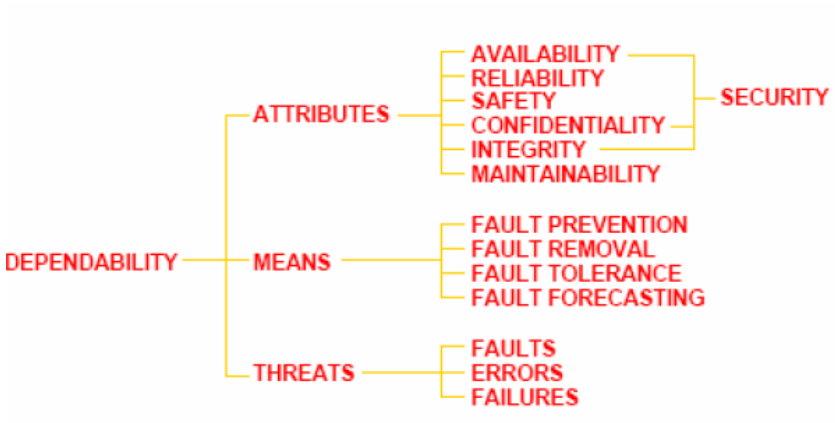
1. Introduction

- In this section, we study **three fundamental characteristics of dependability**:
  - **Attributes (صفات)**: Dependability *attributes* describe the properties which are required from a system.
    - خصوصیت‌های مورد نیاز یک سیستم
  - **Impairment (آسیب‌رساننده‌ها):** Dependability *impairments* express the reasons for a system to cease to perform its function or, in other words, the **threats** to dependability.
    - دلایل توقف اجرای وظایف سیستم یا تهدیدهای اتکاءپذیری
  - **Means (ابزارها):** Dependability *means* are the methods and techniques enabling the development of a dependable computing system.
    - روشها و فنون ساخت یک سیستم کامپیوتری اتکاءپذیر

2.2 Dependability Attributes

- The **attributes** of dependability express the properties which are expected from a system.
- Three primary attributes are
  - **reliability (قابلیت اطمینان),**
  - **availability (قابلیت دسترسی)** and
  - **safety (ایمنی).**
- Other possible attributes include
  - **maintainability (قابلیت نگهداشت),**
  - **testability (آزمون‌پذیری),**
  - **performability (انجام‌پذیری),**
  - **confidentiality (محرمانگی),**
  - **security (امنیت).**
- Depending on the application, one or more of these attributes are needed to appropriately evaluate the system behavior.

دسته‌بندی ارائه‌شده در [ALRL]



2.2 Dependability Attributes

- For example, in an **automatic teller machine (ATM)**:
  - the proportion of time which system is able to **deliver its intended level of service** (system availability) is an important measure.
- For a **cardiac patient (بیمار قلبی) with a pacemaker**:
  - continuous functioning of the device is a matter of life and death.
  - Thus, the ability of the system to deliver its service **without interruption** (system reliability) is crucial.
- In a **nuclear power plant control system**:
  - the ability of the system to perform its functions correctly or to discontinue its function in a **safe** manner (system safety) is of greater importance.

2.1 Reliability

- *Reliability*, $R(t)$, of a system at time t is the probability that the system operates without failure in the interval $[0, t]$, given that the system was performing correctly at time 0.

  - **قابلیت اطمینان** احتمال شرطی این است که سیستم در بازه زمانی [0, t] به درستی کار کند، مشروط بر این که سیستم در ابتدای بازه (0 یا t0) درست بوده باشد.

2.1 Reliability

- **Reliability is a measure of the continuous delivery of correct service.**

  - **پیوستگی و تداوم سرویس درست!**

- **High reliability** is required in situations when a system is expected to operate without interruptions, as in the case of:
  - a **pacemaker**, or
    - اگر سرویس‌دهی دستگاه متوقف شود فرصت تماس با تعمیرکار وجود ندارد!؟!؟!
  - when maintenance cannot be performed because the system cannot be accessed.
    - For example, spacecraft mission control system is expected to provide **uninterrupted service**.
    - مثال این نوع کاربردها، سفینه‌های فضایی هستند (نظیر از مدار خارج شدن سفینه Lewis در سال ۱۹۹۷). امکان اعزام تعمیرکار وجود ندارد!

2.1 Reliability

- **Reliability is a function of time.**
- The way in which time is specified varies considerably depending on the nature of the system under consideration.
  - روش تعیین زمان وابسته به طبیعت سیستم‌های مورد نظر است.
  - For example, if a system is expected to complete its mission in a certain period of time, like in case of a spacecraft, time is likely to be defined as a calendar time or as a number of hours.
  - For software, the time interval is often specified in so called *natural or time units*.
  - A natural unit is a unit related to the amount of processing performed by a software-based product, such as pages of output, transactions, telephone calls, jobs or queries.

2.2 Availability

- **Relatively few systems are designed to operate continuously without interruption and without maintenance of any kind.**
  - تعداد کمی از سیستم‌ها هستند که طوری طراحی می‌شوند که بدون وقفه و بدون هرگونه نگهداشتی به‌طور پیوسته به عملیات ادامه دهند.
- In many cases, we are interested not only in the **probability of failure**, but also in the **number of failures** and, in particular, in the **time required to make repairs**.
- For such applications, attribute which we would like to **maximize is the fraction of time that the system is in the operational state**, expressed by availability.

2.2 Availability

- *Availability*, $A(t)$, of a system at time t is the probability that the system is functioning correctly at the instant of time t.
  - احتمال این است که سیستم در لحظه t به‌درستی در حال کار باشد.
  - قابلیت اطمینان در بازه زمانی تعریف می‌شود ولی قابلیت دسترسی در لحظه زمان.
- $A(t)$ is also referred as *point* availability, or *instantaneous* availability.

2.2 Availability

- Often it is necessary to determine the *interval* or *mission* availability. It is defined by

$$A(T) = \frac{1}{T}\int_0^T A(t)dt. \qquad (2.1)$$

- $A(T)$ is the value of the point availability averaged over some interval of time T.
- This interval might be the life-time of a system or the time to accomplish some particular task.

2.2 Availability

- Finally, it is often found that after some initial transient effect, the point availability assumes a time-independent value.
- In this case, the *steady-state* availability is defined by

$$A(\infty) = \lim_{T \to \infty} \frac{1}{T} \int_0^T A(t)dt. \qquad (2.2)$$

- اگر یک سیستم قابل تعمیر نباشد رابطه R(t) و A(t) چیست؟

2.2 Availability

- If a system cannot be repaired, the point availability $A(t)$ equals to the system's reliability, i.e. the probability that the system has not failed between 0 and t.
- Thus, as T goes to infinity, the steady-state availability of a non-repairable system goes to zero

$$A(\infty) = 0$$

2.2 Availability

- Steady-state availability is often specified in terms of *downtime per year*.
- Table 2.1 shows the values for the availability and the corresponding downtime.

Availability	Downtime
90%	36.5 days/year
99%	3.65 days/year
99.9%	8.76 hours/year
99.99%	52 minutes/year
99.999%	5 minutes/year
99.9999%	31 seconds/year

2.2 Availability

- Availability is typically used as a measure for systems where short interruptions can be tolerated.
- Networked systems, such as telephone switching and web servers, fall into this category.
- A customer of a telephone system expects to complete a call without interruptions.
- However, a downtown of three minutes a year is considered acceptable.

2.2 Availability

- Surveys show that web users lose patience when web sites take longer than eight seconds to show results.
  - البته در آمارهای آخر سال ۲۰۰۶ این زمان به چهار ثانیه رسیده است!
- This means that such web sites should be available all the time and should respond quickly even when a large number of clients concurrently access them.

2.2 Availability

- Another example is **electronic power control system**.
- Customers expect power to be available 24 hours a day, every day, in any weather condition.
- In some cases, prolonged power failure may lead to health hazard (مخاطره سلامتی), due to the loss of services such as water pumps, heating, light, or medical attention.
- Industries may suffer substantial financial loss.

2.3 Safety

- **Safety can be considered as an extension of reliability, namely a reliability with respect to failures that may create safety hazards (مخاطرات).**
- From reliability point of view, all failures are **equal**.
- In case of safety, **failures** are partitioned into:
  - *fail-safe* and
    - **ایمن به خرابی:** اتفاق بدی با وقوع این نوع خرابی نمی‌افتد.
  - *fail-unsafe*.
    - **ناایمن به خرابی:** اتفاق بدی با وقوع این نوع خرابی می‌افتد.

2.3 Safety

- As an example consider an alarm system.
  - The alarm may either **fail to function** even though a dangerous situation exists, or
    - *This is classified as a **fail-unsafe** failure.*
  - It may give a ***false alarm*** when no danger is present.
    - *This is considered a **fail-safe** failure.*

2.3 Safety

- More formally, safety is defined as follows.
  - Safety $S(t)$ of a system is the probability that the system will either perform its function correctly or will **discontinue** its operation in a **fail-safe manner**.
- Safety is required in ***safety-critical applications*** were a failure may result in an human injury, loss of life or environmental disaster.
  - Examples are chemical or nuclear power plant control systems, aerospace and military applications.

2.3 Safety

- **Many unsafe failures are caused by human mistakes.**
  - **مثال:** نیروگاه اتمی چرنوبیل که انجام آزمایشها در شرایطی که سیستم‌های ایمنی به‌طور دستی خاموش شده بودند باعث وقوع فاجعه شد. یک مهندس برق که آشنایی کافی با نیروگاه‌های هسته‌ای نداشت می‌خواست ببیند که می‌توان از انرژی رسوب شده (residual energy) برق تولید نمود یا نه. در نتیجه برخی سیستم‌های ایمنی را خاموش کرده بود که باعث شد فاجعه رخ بدهد...

3. Dependability Impairments (آسیب‌رساننده‌ها)

- Dependability impairment are usually defined in terms of **faults**, **errors**, **failures**.
- معادل‌های فارسی مناسب برای سه اصطلاح فوق؟

3. Dependability Impairments (آسیب‌رساننده‌ها)

- معادل‌های فارسی:
  - **Fault:** خطا
  - **Error:** اشکال
  - **Failure:** خرابی
- مشابهت‌های سه اصطلاح؟

3. Dependability Impairments (آسیب‌رساننده‌ها)

- مشابهت‌های سه اصطلاح:
  - A common feature of the three terms is that **they give us a message that something went wrong**.
- تفاوت‌های سه اصطلاح؟

3. Dependability Impairments (آسیب‌رساننده‌ها)

- تفاوت‌های سه واژه؟
  - A difference is that, in case of a fault, the problem occurred **on the physical level**;
  - In case of an error, the problem occurred **on the computational level**;
  - In case of a failure, the problem occurred **on a system level**.
  - برای مثال یک بیت حافظه ممکن است به‌طور دائمی یک شده باشد: **stuck-at-1 fault**
  - در زمان اجرای یک برنامه خطای فوق باعث می‌شود که یک متغیر برنامه که در همان بخش خطادار حافظه نگهداری می‌شود مقدار درستی نداشته باشد: **Error**
  - و اگر این برنامه یک سیستم را کنترل می‌کند، یک خرابی اتفاق بیفتد: **Failure**

3.1 Faults, errors and failures

- **A *fault* is a physical defect (عیب), imperfection (نقص), or flaw (کاستی) that occurs in some hardware or software component.**
  - Examples are short-circuit between two adjacent interconnects, broken pin, or a software bug.

3.1 Faults, errors and failures

- **An *error* is a deviation from correctness or accuracy in computation, which occurs as a result of a fault.**
  - اشکال یک انحراف از درستی یا دقت در محاسبه است که در نتیجه خطا حادث می‌شود.
- **Errors are usually associated with incorrect values in the system state.**
  - For example,
    - a circuit or a program computed an incorrect value,
    - an incorrect information was received while transmitting data.

3.1 Faults, errors and failures

- **A *failure* is a non-performance of some action which is due or expected.**
  - خرابی عدم اجرای برخی عملیات در موعد مقرر و مطابق انتظار.
- A system is said to have a failure if the service it delivers to the user deviates from compliance with the system specification for a specified period of time.
- A system may fail either:
  - because it does not act **in accordance with the specification**, or
  - because the **specification did not adequately describe its function**.

3.1 Faults, errors and failures

- **Faults are reasons for errors and errors are reasons for failures.**
  - **Faults => Errors => Failures**
- For example, consider a power plant, in which a computer controlled system is responsible for monitoring various plant temperatures, pressures, and other physical characteristics.
  - سنسور به **خطا** گزارش می‌کند که سرعت توربین کاهش یافته است.
  - خطای فوق باعث می‌شود که سیستم بخار بیشتری را از آنچه لازم است به توربین ارسال کند (error) که یا باعث افزایش زیادی سرعت توربین می‌شود.
  - سیستم ایمنی مکانیکی برای جلوگیری از آسیب، توربین را خاموش می‌کند. در نتیجه سیستم دیگر برق تولید نمی کند (خرابی سیستم، ایمن به خرابی).

3.1 Faults, errors and failures

- خطا، اشکال و خرابی و همین طور سطح فیزیکی، محاسباتی و سیستمی در مورد نرم‌افزار چگونه هستند؟

3.1 Faults, errors and failures

- Definitions of physical, computational and system level are a bit more confusing when applied to software.
  - We interpret **a program code** as **physical level**,
  - **the values of a program state** as **computational level**, and
  - **the software system running** the program as **system level**.
- For example, an operating system is a software system.
  - Then, a bug in a program is a fault,
  - possible incorrect value caused by this bug is an error and
  - possible crush of the operating system is a failure.

3.1 Faults, errors and failures

- Not every fault cause error and not every error cause failure.
- This is particularly evident in software case.
- Some program bugs are very hard to find because they cause failures only in very specific situations.
  - For example, in November 1985, $32 billion overdraft (حواله بیش از اعتبار) was experienced by the Bank of New York, leading to a loss of $5 million in interests.
  - The failure was caused by an unchecked overflow of an 16-bit counter.

3.1 Faults, errors and failures

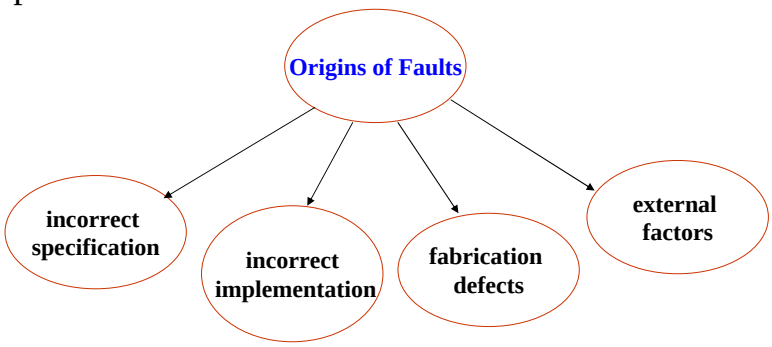
- In 1994, Intel Pentium I microprocessor was discovered to compute incorrect answers to certain floating-point division calculations.
- For example, dividing 5505001 by 294911 produced 18.66600093 instead of 18.66665197.
- The problem had occurred because of the omission of five entries in a table of 1066 values used by the division algorithm.
- The five cells should have contained the constant +2, but because the cells were empty, the processor treated them as a zero.

3.2 Origins of faults (منابع خطا)

- As we discussed earlier, failures are caused by errors and errors are caused by faults.
- Faults are, in turn, caused by **numerous problems** occurring at **specification**, **implementation**, **fabrication stages of the design process**.
- They can also be caused by **external factors**, such as **environmental disturbances** (اختلالات محیطی) or **human actions**, either **accidental or deliberate** (عمدی).

3.2 Origins of faults (منابع خطا)

- We can classify the **sources of faults** into four groups:



3.2 Origins of faults (منابع خطا)

- *Incorrect specification* results from incorrect **algorithms**, **architectures**, or **requirements**.
- Faults caused by incorrect specifications are usually called ***specification faults***.

3.2 Origins of faults (منابع خطا)

- In System-on-a-Chip (SoC) design, integrating pre-designed intellectual property (IP) cores (هسته‌های مالکیت معنوی), specification faults are one of the most common type of faults.
  - Core specifications, provided by the core vendors, **do not always contain** all the details that system-on-a-chip designers need.
  - This is partly due to the intellectual property protection requirements, especially for core netlists and layouts.

3.2 Origins of faults (منابع خطا)

- Faults due to ***incorrect implementation***, usually referred to as ***design faults***, occur when the system implementation does not adequately implement the specification.

- به خطاهایی که به سبب پیاده‌سازی نادرست ایجاد می‌شوند، خطاها طراحی گفته می‌شود و زمانی رخ می‌دهند که پیاده‌سازی سیستم به اندازه کافی با مشخصاتش (توصیفش) مطابقت نداشته باشد.

3.2 Origins of faults (منابع خطا)

- Incorrect implementation in hardware include
  - poor component selection,
  - logical mistakes,
  - poor timing or synchronization.

3.2 Origins of faults (منابع خطا)

- In software, examples of incorrect implementation are
  - bugs in the program code and
  - poor software component reuse.
- Software heavily relies on different **assumptions** about its **operating environment**.
- Faults are likely to occur if these assumptions are incorrect in the new environment.
  - مثال موشک آریان در اسلاید بعد...

3.2 Origins of faults (منابع خطا)

- The Ariane 5 rocket accident is an example of a failure caused by a reused software component.
- Ariane 5 rocket exploded 37 seconds after lift-off on June 4th, 1996, because of a software fault that resulted from converting a 64-bit floating point number to a 16-bit integer.
- The value of the floating point number happened to be larger than the one that can be represented by a 16-bit integer.
- In response to the overflow, the computer cleared its memory.
- The memory dump was interpreted by the rocket as an instruction to its rocket nozzles (دماغه‌ها), which caused an explosion.

3.2 Origins of faults (منابع خطا)

- A source of faults in hardware are ***component defects***.
- These include
  - manufacturing imperfections (نواقص مرحله ساخت و تولید مولفه),
  - random device defects (نواقص دستگاه که به‌طور تصادفی رخ می‌دهند؟!؟!) and
  - components wear-outs (مولفه‌ها (به دلیل عمر زیاد) فرسودگی).

3.2 Origins of faults (منابع خطا)

- **Fabrication defects** were the primary reason for applying **fault-tolerance techniques** to early computing systems, due to the low reliability of components.
- Following the development of semiconductor technology, hardware components became intrinsically (به‌طور اساسی) more reliable and the percentage of faults caused by fabrication defects diminished.

3.2 Origins of faults (منابع خطا)

- The fourth cause of faults are ***external factors***, which arise from
  - outside the system boundary,
  - the environment,
  - the user or the operator.

3.2 Origins of faults (منابع خطا)

- External factors include phenomena that directly affect the operation of the system, such as temperature, vibration, electrostatic discharge, nuclear or electromagnetic radiation or that affect the inputs provided to the system.
- For instance, radiation causing a bit to flip (معکوس شود) in a memory location is a fault caused by an external factor.

3.2 Origins of faults (منابع خطا)

- Faults caused by user or operator mistakes can be **accidental** or **malicious** (از روی عناد).
- For example, a user can accidentally provide incorrect commands to a system that can lead to system failure,
  - e.g. improperly initialized variables in software.
- Malicious faults are the ones caused, for example, by software viruses and hacker intrusions.

3.3 Common-mode faults

- A ***common-mode fault*** is a fault which occurs simultaneously in two or more redundant components.
- Common-mode faults are caused by phenomena that create **dependencies** between the redundant units which cause them to fail simultaneously,
  - i.e. common communication buses or shared environmental factors.
- Systems are **vulnerable** to common-mode faults if they rely on a single source of power, cooling or input/output (I/O) bus.

3.3 Common-mode faults

- Another possible source of common-mode faults is a design fault which causes redundant copies of hardware or of the same software process to fail under identical conditions.
- The only fault-tolerance approach for combating common-mode design faults is **design diversity** (تنوع طراحی).
- ***Design diversity*** is the implementation of more than one variant of the function to be performed.

3.4 Hardware faults

- Hardware faults are classified with respect to **fault duration** into:
  - **permanent** (دائمی),
  - **transient** (گذرا) and
  - **intermittent** (نوبتی، متناوب) faults.

3.4 Hardware faults

- A ***permanent fault*** (خطای دائمی) remains active until a **corrective action** is taken.
- These faults are usually caused by some **physical defects** in the hardware, such as
  - **shorts in a circuit**,
  - **broken interconnect** or
  - a **stuck bit (بیت متصل)** in the memory:
    - **بیتی که به‌طور دائمی ۱ یا صفر شده است: stuck-at-1 یا stuck-at-0**
- Permanent faults can be detected by **on-line test routines** that work concurrently with normal system operation.

3.4 Hardware faults

- A ***transient fault*** (خطای گذرا) remains active for a short period of time.
  - Because of their short duration, transient faults are often detected through the errors that result from their propagation.
- Transient faults are often called *soft faults* (خطاهای نرم) or *glitches* (اشکال).
- **Transient fault are dominant** (عمده) **type of faults in computer memories.**
  - For example, about 98% of RAM faults are transient faults.

3.4 Hardware faults

- A transient fault that becomes active periodically is an ***intermittent fault*** (خطای متناوب).
- **Intermittent faults** can be due to:
  - implementation flaws (ایرادهای پیاده‌سازی), aging (پیری) and wear-out (فرسودگی), and
  - unexpected operation environment (محیط عملیاتی غیرمنتظره).

مدلهای خطا

- امکان برشمردن همه انواع خطاهایی که در یک سیستم ممکن است رخ دهند وجود ندارد. از اینرو برای ارزیابی امکان‌پذیری **پوشش خطا** (fault coverage)، فرض می‌شود که خطاها مطابق **مدلهای خطا** (fault models) رفتار می‌کنند.
  - **یک مدل خطا سعی می‌کند که اثر خطایی را که ممکن است رخ دهد توصیف نماید.**
- **عمومی‌ترین مدلهای خطا عبارتند از:**
  - **خطای اتصال** (stuck-at)
  - **خطای گذر** (transition fault)
  - **خطای پیوستگی** (coupling fault)

خطای اتصال

- **یک خطای اتصال (stuck-at fault) باعث می‌شود که یک خط (سیم) مدار یا یک سلول حافظه به‌طور دائمی دارای یک مقدار منطقی یک یا صفر شود.**
- فرض می‌شود که وظیفه مبنایی مدار با خطا تغییر نمی‌کند.
  - یعنی مثلاً یک گیت AND به دلیل وقوع این نوع خطا به یک گیت OR تبدیل نمی‌شود.
- **به دلیل سادگی و تاثیر قابل توجه، خطای اتصال عمومی‌ترین مدل خطا است.**

خطای گذر

- **یک خطای گذر (transition fault) خطایی است که یک خط مدار نمی‌تواند از یک حالت خاص به حالتی دیگر تغییر نماید.**
  - برای مثال فرض کنید که یک سلول حافظه شامل یک مقدار صفر است. اگر یک مقدار یک در آن سلول نوشته شود، سلول به‌طور موفقیت‌آمیزی حالتش را تغییر می‌دهد.
  - اما اگر متعاقباً یک مقدار صفر در سلول نوشته شود، حالت سلول تغییر نمی‌کند. (یعنی اگر از حافظه خوانده شود، همان مقدار قبلی یک را خواهد داشت.)
  - در این صورت گفته می‌شود که حافظه دارای **خطای گذر یک-به-صفر** (one-to-zero transition fault) دارد.
- **هم خطاهای اتصال و هم خطاهای گذر به آسانی در طی تست قابل تشخیص هستند.**

خطاهای پیوستگی

- **خطاهای پیوستگی** (coupling faults) مشکل‌ترین خطاها برای آزمون هستند. زیرا به بیش از یک خط (سیم) بستگی دارند.
- مثالی از یک خطای پیوستگی می‌تواند یک اتصال کوتاه (short-circuit) بین خطوط دو کلمه مجاور حافظه باشد. در نتیجه این خطا نوشتن مقداری در یکی از دو کلمه باعث نوشتن همان مقدار در کلمه مجاور خواهد شد.
- **دو نوع خطای پیوستگی عبارتند از:**
  - **خطاهای پیوستگی معکوس** (inversion coupling faults): یک گذر در یک سلول حافظه محتوای سلول مجاور را معکوس می‌کند.
  - **خطاهای پیوستگی همانی** (idempotent coupling faults): یک گذر خاص یک سلول حافظه باعث می‌شود که یک مقدار خاص (صفر یا یک) در سلول دیگر نوشته شود.

خطاهای پیوستگی

- **واضح است که مدلهای خطا در ۱۰۰٪ موارد دقیق نیستند. زیرا خطاها می‌توانند باعث اثرات متنوعی شوند.**
- اما مطالعات نشان داده است که ترکیبی از چند مدل خطا می‌تواند پوشش خیلی دقیقی از خطاهای واقعی را فراهم نماید.
- برای مثال در مورد حافظه‌ها عملاً همه خطاها می‌توانند به‌صورت ترکیبی از خطاهای چسپیدگی، خطاهای گذر ، و خطاهای پیوستگی همانی مدل شوند.

۳-۵ خطاهای نرم‌افزاری

- **نرم‌افزار از چه جنبه‌هایی با سخت‌افزار متفاوت است؟**
- **نرم‌افزار از جنبه‌های زیر با سخت‌افزار متفاوت است:**
  - **اول: نرم‌افزار پیر نشده یا فرسوده نمی‌شود.**
  - **دوم: نرم‌افزار در طول مدت چرخه زندگی سیستم قابل ارتقاء (upgrade) است.**
  - **سوم: رفع اشکالات (bugs) لزوماً منجر به افزایش قابلیت اطمینان نرم‌افزار نمی‌شود.**
  - **سرانجام: از آنجایی که نرم‌افزار ذاتاً پیچیده‌تر بوده و دارای قاعده‌مندی کمتری نسبت به سخت‌افزار است، حصول پوشش درستی‌یابی کافی، مشکل‌تر است.**

۳-۵ خطاهای نرم‌افزاری

- **اول: نرم‌افزار پیر نشده یا فرسوده نمی‌شود.**
  - برخلاف قطعات مکانیکی یا الکترونیکی سخت‌افزار، نرم‌افزار کج یا شکسته نشده یا بوسیله عوامل محیطی تحت تاثیر قرار نمی‌گیرد.
  - با فرض نمودن اینکه نرم‌افزار قطعی است (و رفتارهای غیرقطعی از خود نشان نمی‌دهد)، همیشه رفتار یکسانی را در مواقع یکسان از خود نشان می‌دهد، مگر آنکه مشکلاتی در سخت‌افزار وجود داشته باشد که محتوی حافظه یا مسیر داده‌ها را تغییر دهد.
  - از آنجایی که از وقتی که نرم‌افزار در حافظه بارگذاری شده و شروع به اجرا نموده است تغییر نمی‌کند، سعی در حصول تحمل‌پذیری خطا بوسیله تکرار ماجول‌های نرم‌افزاری یکسان بی فایده است. **زیرا همه نسخه‌ها دارای خطاهای یکسانی خواهند بود.**

۳-۵ خطاهای نرم‌افزاری

- **دوم آنکه: نرم‌افزار در طول مدت چرخه زندگی سیستم قابل ارتقاء (upgrade) است.**
- این ارتقاء می‌تواند یک **ارتقاء اطمینان** (reliability upgrade) یا **ارتقاء قابلیت** (feature upgrade) باشد.
  - **ارتقاء اطمینان** با هدف بهبود قابلیت اطمینان یا امنیت نرم‌افزار انجام می‌شود. این کار اغلب با طراحی مجدد یا پیاده‌سازی مجدد برخی از ماجول‌های نرم‌افزاری با رهیافت‌های مهندسی بهتر انجام می‌شود.
  - **ارتقاء قابلیت** با هدف بهبود عملکرد (functionality) نرم‌افزار انجام می‌شود. این کار منجر به افزایش پیچیدگی و لذا کاهش قابلیت اطمینان بوسیله معرفی (introduction) محتمل خطاهای جدید به نرم‌افزار می‌شود.

۳-۵ خطاهای نرم‌افزاری

- **سوم: رفع اشکالات (bugs) لزوماً منجر به افزایش قابلیت اطمینان نرم‌افزار نمی‌شود.**
  - برعکس مشکلات جدید غیرمنتظره ممکن است بروز کنند.
  - برای مثال در سال ۱۹۹۱ یک تغییر در تنها سه خط کد در یک برنامه سیگنالینگ (مخابراتی) که دارای میلیونهای خط کد بود، منجر به متوقف شدن سیستم‌های تلفنی کالیفرنیا و کل سواحل شرقی ایالات متحده شد.

۳-۵ خطاهای نرم‌افزاری

- **سرانجام: از آنجایی که نرم‌افزار ذاتاً پیچیده‌تر بوده و دارای قاعده‌مندی کمتری نسبت به سخت‌افزار است، حصول پوشش درستی‌یابی کافی مشکل‌تر است.**
  - روشهای سنتی آزمون و اشکال‌زدایی برای سیستم‌های نرم‌افزاری بزرگ کافی نیستند.
  - تمرکز در سالهای اخیر بر روی **روشهای صوری** نوید پوشش بالاتر را می‌دهد، اما این روشها به سبب پیچیدگی محاسباتی (!؟!؟) خیلی زیاد، تنها در کاربردهای خاص قابل به‌کارگیری هستند.
  - به سبب درستی‌یابی ناکافی، اغلب خطاهای نرم‌افزاری، **خطاهای طراحی** (design faults) هستند و وقتی رخ می‌دهند که یا یک برنامه‌ساز توصیف مشخصات را اشتباه درک می‌کند (misunderstands) یا آنکه به سادگی یک اشتباه (mistake) انجام می‌دهد.
  - خطاهای طراحی به عوامل انسانی فازی (نادقیق) مرتبطند و بنا بر این اجتناب از آنها سخت‌تر است.
  - در سخت‌افزار، خطاهای طراحی ممکن است وجود داشته باشند، اما سایر انواع خطاها، نظیر عیوب ساخت و خطاهای گذرای ایجاد شده توسط عوامل محیطی اغلب مهمتر هستند.

ابزارها اتکاءپذیری

- **ابزارهای اتکاءپذیری (dependability means)، روشها و فنونی هستند که ساخت یک سیستم اتکاءپذیر را میسر می‌کنند.**
- **ابزارهای اصلی اتکاءپذیری عبارتند از:**
  - **تحمل‌پذیری خطا** (fault tolerance)
  - **اجتناب از خطا** (fault prevention)
  - **رفع خطا** (fault removal)
  - **پیش‌بینی خطا** (fault forecasting)

ابزارها اتکاءپذیری

- **تحمل‌پذیری خطا** یکی از مهم‌ترین روشهایی است که در کنار روشهای دیگر برای حصول اتکاءپذیری استفاده می‌شود.
- هدف **اجتناب از خطا**، جلوگیری از وقوع خطاها است.
- هدف **رفع خطا**، کاهش تعداد خطاهایی است که در سیستم وجود دارند.
- اما هدف **پیش‌بینی خطا**، تخمین آن است که:
  - چه تعداد خطا وجود دارند؟
  - وقوع خطاهای محتمل در آینده چگونه است؟ و
  - تاثیر خطاها بر سیستم چیست؟

تحمل‌پذیری خطا

- **هدف تحمل‌پذیری خطا ساخت سیستم‌هایی است که در حضور خطاها به‌درستی عمل می‌کنند.**
- تحمل‌پذیری خطا با استفاده از انواعی از **افزونگی** (redundancy) حاصل می‌شود.
- افزونگی فراهم‌سازی برخی قابلیت‌های عملکردی است که در یک محیط عاری از خطا مورد نیاز نیستند.
- افزونگی اجازه می‌دهد که یک خطا **پوشانده شود** (mask) یا آنکه یک خطا **کشف** (detect) شده و متعاقباً **مکان‌یابی** (location)، **محدودسازی(تحدید)** (containment) و **بازیابی** (recovery) شود.

تحمل‌پذیری خطا

- **پوشاندن خطا (fault masking) فرایند تضمین آن است که علی رغم حضور خطا تنها مقادیر درست به خروجی سیستم منتقل شوند.**
  - این کار با اجتناب از اینکه سیستم بوسیله اشکالات (errors) تحت تاثیر قرار بگیرد انجام می‌شود و برای این منظور یا اشکالات تصحیح شده یا آنکه به طریقی جبران می‌شوند (compensate).
  - از آنجایی که سیستم اثر خطا را نشان نمی‌دهد، در نتیجه وجود خطا برای کاربر یا اپراتور نامرئی است.
  - برای مثال، حافظه‌ای که بوسیله یک **کد تصحیح اشکال** (error-correcting code) محافظت شده است، بیتهای خطادار را قبل از آنکه سیستم از داده‌ها استفاده کند تصحیح می‌کند.
  - مثال دیگر از پوشاندن خطا در **افزونگی ماجولار سه‌گانه** (TMR: triple module redundncy) مبتنی بر رای اکثریت است.

تحمل‌پذیری خطا

- **کشف خطا (fault detection) فرایند تعیین آن است که یک خطا در یک سیستم اتفاق افتاده است.**
- مثالهایی از فنون کشف خطا عبارتند از **آزمون پذیرش** (acceptance test) و **مقایسه**.
  - **آزمونهای پذیرش** در پردازنده‌ها مرسوم هستند. نتیجه یک برنامه، مورد آزمون قرار می‌گیرد. اگر نتیجه از آزمون عبور کند (pass)، اجرای برنامه ادامه می‌یابد. عدم عبور از آزمون به معنای وجود خطا است.
  - **مقایسه** در سیستم‌هایی که دارای مولفه‌های تکرار شده هستند استفاده می‌شود. ناسازگاری (disagreement) در نتایج، وجود خطا را مشخص می‌کند.

تحمل‌پذیری خطا

- ***Fault location*** **is the process of determining where a fault has occurred.**
  - A failed acceptance test cannot generally be used to locate a fault. It can only tell that something has gone wrong.
  - Similarly, when a disagreement occurs during comparison of two modules, it is not possible to tell which of the two has failed.

تحمل‌پذیری خطا

- ***Fault containment*** **is the process of isolating a fault and preventing propagation of the effect of that fault throughout the system.**
  - The purpose is to limit the spread of the effects of a fault from one area of the system into another area.
  - This is typically achieved by frequent fault detection (کشف خطای مکرر), by multiple request/confirmation protocols and by performing consistency checks between modules.

تحمل‌پذیری خطا

- Once a faulty component has been identified, a system *recovers* by reconfiguring itself to isolate the component from the rest of the system and regain operational status.
  - This might be accomplished by having the component replaced, by marking it off-line and using a redundant system.
  - Alternately, the system could switch it off and continue operation with a degraded capability.
  - This is known as *graceful degradation* (تنزل آبرومندانه).

اجتناب از خطا

- **Fault prevention** is achieved by quality control techniques during specification, implementation and fabrication stages of the design process.
  - For hardware, this includes **design reviews**, **component screening** and **testing**.
  - For software, this includes **structural programming**, **modularization** and **formal verification techniques**.

اجتناب از خطا

- A rigorous design review may eliminate many of the specification faults.
- If a design is efficiently tested, many of design faults and component defects can be avoided.
- Faults introduced by external disturbances such as lightning or radiation are prevented by shielding (پوشش‌دار کردن), radiation hardening (مقاوم کردن), etc.
- User and operation faults are avoided by training and regular procedures for maintenance.
- Deliberate malicious faults (خطاهای زیان‌رسان عمدی) caused by viruses or hackers are reduced by firewalls or similar security means.

رفع خطا

- **Fault removal is performed during the development phase as well as during the operational life of a system.**
  - During the development phase, fault removal consists of three steps: **verification**, **diagnosis** and **correction**.
  - Fault removal during the operational life of the system consists of **corrective and preventive maintenance**.
- *Verification* is the process of checking whether the system meets a set of given conditions.
  - If it does not, the other two steps follow: the fault that prevents the conditions from being fulfilled is diagnosed and the necessary corrections are performed.

انواع نگهداشت

- In *preventive maintenance*, parts are replaced, or adjustments are made before failure occurs.
- The objective is to increase the dependability of the system over the long term by staving off (دفع کردن) the aging effects of wear-out.
- In contrast, *corrective maintenance* is performed after the failure has occurred in order to return the system to service as soon as possible.

پیش‌بینی خطا

- **Fault forecasting is done by performing an evaluation of the system behavior with respect to fault occurrences or activation.**
  - Evaluation can be *qualitative*, that aims to rank the failure modes or event combinations that lead to system failure, or *quantitative*, that aims to evaluate in terms of probabilities the extent to which some attributes of dependability are satisfied, or **coverage**.
  - Informally, *coverage* is the probability of a system failure given that a fault occurs.
  - Simplistic estimates of coverage merely measure redundancy by accounting for the number of redundant success paths in a system.
  - More sophisticated estimates of coverage account for the fact that each fault potentially alters a system's ability to resist further faults.
  - **We study qualitative and quantitative evaluation techniques in more details in the next section.**

Assignment #1

- **Problems:**
  - **2.2**
  - **2.8**
  - **2.10**
  - **2.15**
  - **2.19**
  - **2.23**
  - **2.28**
- **Due: 86/2/1**

- آقای مومنی روز شنبه ۸۶/۱/۲۵ باقیمانده مقاله [ALRL] را ارائه نمایند.