Chapter 11 Notes

The dependability of systems is now usually more important than their described functionality for the following reasons:

* System failures affect a large number of people
* Users often reject systems that are unreliable, unsafe, or insecure (and furthermore, they may reject any other systems from the same company)
* System failure costs may be enormous (i.e. crashing Wall Street, Air Traffic Control)
* Undependable systems may cause information loss

When designing a dependable system, you must consider:

* Hardware Failure
  + System hardware may fail because of mistakes in its design because components fail as a result of manufacturing errors, or because components have reached the end of their natural life
* Software Failure
  + System software may fail because of mistakes in it specification, design, or implementation
* Operational Failure
  + Human users may fail to use or operate the system correctly.

**Dependability Properties (Section 11.1)**

* Dependability of a computer system is a property of the system that reflects its trustworthiness.
* Trustworthiness means the degree of confidence a user has that the system will operate as they expect, and that the system will not “fail” in normal use.
* Principal Dependability Properties
  + Availability
    - The ability of the system to deliver services when requested
    - The probability that it will be up and running and able to deliver useful services to users at any given time
  + Reliability
    - The ability of the system to deliver servies as specified
    - The probability that the system will correctly deliver services as expected by the user
  + Safety
    - The ability of the system to operate without catastrophic failure
    - How likely it is that the system will cause damage to people or its environment
  + Security
    - The ability of the system to protect itself against accidental or deliberate intrusion
    - How likely it is that the system can resist accidental or deliberate intrusions
* Other dependability properties
  + Repairability
    - Can the system be repaired quickly if it fails?
  + Maintainability
    - Can the system adapt to new requirements?
  + Survivability
    - Can the system continue to deliver service whilst under attack and while it is disabled
  + Error tolerance
    - Can the system detect errors and either fix them automatically or request the user to reinput their data?
* To develop dependable SW, you need to ensure that:
  + You avoid the introduction of accidental errors into the system during SW specification and development
  + You design verification and validation processes that are effective in discovering residual errors that affect the dependability of the system
  + You design protection mechanisms that guard against external attacks that can compromise the availability or security of the system
  + You configure the deployed system and its supporting SW correctly for its operating environment
* In addition you should include recovery mechanisms that make it possible to restore normal system service as quickly as possible.
* Increasing the dependability of a system increases the development costs
  + This is due to the extra design, implementation, and validations costs
  + Testing is very expensive

**Availability and Reliability (Section 11.2)**

* These are 2 related properties
* Availability = Probability that the system will be up and running to deliver these services to users on request.
* Reliability = Probability that the system’s services will be delivered as defined in the system specification
* More precise definitions
  + Reliability =Probability of failure-free operation over a specified time, in a given-environment, for a specific purpose
  + Availability = Probability that a system, at a point in time, will be operational and able to deliver the requested services.
* A common cause of perceived unreliability is that the system specification does not match the expectations of the system users
* Availability not only depends on the number of system crashes, but also on the time need to repair faults that have cause the failure.
* Reliability Terminology
  + Human error or mistakes
    - Human behavior that results in the introduction of faults into a system. For example, in the wilderness weather system, a programmer might decide that the way to compute the time for the next transmission is to add 1 hour to the current time. This works expect when the transmission time is between 23:0 and midnight (24:00)
  + System Fault
    - A characteristic of a system that can lead to a system error. The fault is the inclusion of the code to add 1 hour to the time of the last transmission, without a check if the time is greater than or equal to 23:00.
  + System Error
    - An erroneous system state that can lead to system behavior that is unexpected by system users. The value of the transmission time is set incorrectly (to 24:xx rather than 00:xx) when the faulty code is executed
  + System Failure
    - An event that occurs at some point when the system does not deliver a service as expected by its users. No weather data is transmitted because the time is invalid.
* When an input or sequence of events causes faulty code in a system to be executed, an erroneous state is created that may lead to a software failure.
* A program reliability depends on the number of system inputs that are members of the set of inputs that lead to an erroneous output
  + The practical reliability of a program depends on the number of inputs causing erroneous outputs (failures) during normal use of the system by most users.
* System faults don’t always lead to system errors and system errors do not always lead to system failures. The reasons are:
  + Not all code in a program is executed
    - The code that includes a fault may never be executed because of the way that the software is used
  + Errors are transient
    - A state variable may have an incorrect value caused by the execution of faulty code. However, before this is accessed and causes a system failure, some other system input may be processed that resets that state to a valid value.
  + The system may include fault detection and protection mechanisms
    - These ensure that the erroneous behavior is discovered and corrected before the system services are affected
  + Users adapt their behavior to avoid using inputs that they know cause program failures
* 3 Approaches to improve reliability
  + Fault Avoidance
    - Use techniques that either minimizes the possibility of human errors and/or that trap mistakes before they result in the introduction of system faults.
  + Fault Detection and Removal
    - Use of Verification and Validation techniques that increase the chances that faults will be detected and removed before the system is used.
    - Ex. Systematic testing and debugging
  + Fault Tolerance
    - Techniques that ensure that faults in system do not result in system errors or that system errors do not result in system failures. Incorporation of self-checking facilities in a system and the use of redundant system modules.

**Safety (Section 11.3)**

* Safety-critical systems are system where it is essential that system operation is always safe
* 2 Classes of Safety Critical SW
  + Primary Safety Critical SW
    - SW that is embedded as a controller in a system
    - Malfunctioning of such SW can cause a hardware malfunction, which results in human injury or environmental damage.
  + Secondary Safety-Critical SW
    - SW that can indirectly result in a injury
    - Ex: Computer-aided engineering design system whose malfunctioning might result in a design fault in the object being designed
* 4 Reasons why Reliable System may NOT be Safe:
  + We can never be 100% certain that a system is fault-free and fault-tolerant. Undetected faults can be dormant for a long time and software failures can occur after many years of reliable operation
  + The specification may be incomplete in that is does not describe the required behavior of the system in some critical situations
    - A high percentage of system malfunctions are the result of specification rather than design errors.
  + Hardware malfunction may cause the system to behave in an unpredictable way, and present the software with an unanticipated environment. When components are close to physical failure, they may behave erratically and generate signals that are outside the ranges that can be handled by the SW.
  + The system operators may generate inputs that are not individually incorrect but which, in some situations, can lead to a system malfunction.
* Safety Terminology
  + Accident/Mishap
    - An unplanned event or sequence of events which results in human death or injury, damage to property, or to the environment.
    - Ex. Overdose of insulin
  + Hazard
    - A condition with the potential for causing or contributing to an accident.
    - Ex. Failure of the sensor that measures blood glucose
  + Damage
    - A measure of the loss resulting from an accident/mishap.
  + Hazard Severity
    - An assessment of the worst possible damage that could result from a particular hazard. Hazard severity can range from catastrophic, where many people are killed, to minor, where only minor damage results.
  + Hazard Probability
    - Probability of the events occurring which create a hazard
  + Risk
    - Measure of the probability that the system will cause an accident.
    - Assessed by considering the Hazard Probability, the Hazard Severity, and the probability that the hazard will lead to an accident.
* The key to assuring safety is to ensure either that accidents do not occur or that the consequences of an accident are minimal. This can be achieved in 3 complementary ways:
  + Hazard Avoidance
    - System is designed to avoid hazards
    - Ex. A cutting system that requires an operator to use 2 hands to press separate buttons simultaneously avoids the hazard of the operator’s hands being in the blade pathway.
  + Hazard Detection and Removal
    - The system is designed so that hazards are detected and removed before they result in an accident
    - Ex. A chemical plant system may detect excessive pressure and open a relief valve to reduce these pressures before an explosion occurs
  + Damage limitation
    - System may include protection features that minimize the damage that may result from an accident.
    - Ex. An aircraft engine normally includes automatic fire extinguishers. If a fire occurs, it can often be controlled before it poses a threat to the aircraft.
* Accidents most often occur when several things go wrong at the same time
* Accidents are an inevitable part of using complex systems

**Security (11.4)**

* Security = Attribute that reflects the ability of the system to protect itself from external attacks (Accidental or Deliberate)
* Security Terminology
  + Asset
    - Something of value which has to be protected. The asset may be the SW system itself or data used by that system
    - Ex. Records of each patient in a database
  + Exposure
    - Possible loss or harm to a computing system. This can be loss or damage to data, or can be a loss of time and effort if recovery is necessary after a security breach.
    - Ex. Potential financial loss from future patients who do not seek treatment because they do not trust the clinic to maintain their data.
  + Vulnerability
    - A weakness in a computer-based system that may be exploited to cause loss or harm
    - Ex. A weak password system which makes it easy for users to set passwords. User ids that are the same as names
  + Attack
    - An exploitation of a system’s vulnerability. Generally, this is from outside the system and is a deliberate attempt to cause some damage
    - Ex. An impersonation of an authorized user
  + Threats
    - Circumstances that have potential to cause loss or harm. You can think of these as a system vulnerability that is subject to an attack.
    - Ex. An unauthorized user will gain access to the system by guessing the credentials (login name and pw) of an authorized user
  + Control
    - A protective measure that reduces a system’s vulnerability. Encryption is an example of a control that reduces a vulnerability of a weak access control system.
    - Ex. A password checking system that disallows user passwords that are proper names or words that are normally included in a dictionary.
* 3 Main Types of Threats in a Networked System
  + Threats to confidentiality of the system and its data
  + Threats to the integrity of the system and its data (i.e. damage or corrupt data)
  + Threats to the availability of the system and its data
* 3 Controls you might use to enhance security
  + Vulnerability Avoidance
    - Controls that are intended to ensure that attacks are unsuccessful
    - Ex. Military System are not connected to public networks so that external access is impossible
  + Attack Detection and Neutralization
    - Controls that are intended to detect and repel attacks. Includes functionality that monitors the system’s operation and checks for unusual patterns of activity.
  + Exposure limitation and recovery
    - Controls that support recovery from problems
    - Ex. Automated backup strategies and information “mirroring” to insurance policies that cover the costs associated with a successful attack on the system