Chapter 12 Notes

System dependability does NOT just depend on good engineering. It also requires attention to detail when the system requirements are derived and the inclusion of special software requirements that are geared to ensuring the dependability and security of the system. Those dependability and security requirements are of 2 types:

* Functional Requirements
  + Define checking and recovery facilities that should be included in the system and features that provide protection against system failures and external attacks
* Non-Functional Requirements
  + Define the required reliability and availability of the system

The starting point for generating functional dependability and security requirements is often high-level business or domain rules, policies, or regulations. These form “shall not” rules.

**Risk-driven requirements specification (Section 12.1)**

* Dependability and security requirements specify how the system should protect itself from internal faults, stop system failures causing damage to the environment, stop accidents or attacks from the system’s environment damaging the system, and facilitate recovery in the event of failure.
* A risk-driven approach is an approach that has been widely used by safety- and security-critical system developers. It focuses on those events that could cause the most damage or that are likely to occur frequently. Events that have only minor consequences or that are extremely rare may be ignored.
  + Safety-critical
    - Risk are associated with hazards that can result in accidents
  + Security-critical
    - Risk come from insider and outsider attacks on a system that are intended to exploit vulnerabilities
* Stages in the process
  + Risk Identification
    - Potential risk are identified
    - Risks are dependent on the environment in which the system is to be used.
  + Risk Analysis and Classification
    - Risks are considered separately
    - Potentially serious risks and those that are not implausible are further analyzed
    - Risks may be eliminated because they are unlikely to arise or because they cannot be detected by the software
  + Risk Decomposition
    - Each risk is analyzed to discover potential root causes of the risk
    - Root causes are the reasons why the system may fail
      * They may be hardware or software or inherent vulnerabilities
  + Risk Reduction
    - Proposals for ways in which the identified risks may be reduced or eliminated are made
* For large system, Risk Analysis may be structured into phases:
  + Preliminary Risk Analysis
    - Major risks from the environment are identified
    - These are independent from the technology being used for system development
    - Goal is to develop an initial set of security and dependability requirements for the system
  + Life-Cycle Risk Analysis
    - Takes place during the system development
    - Concerned mostly with risks that arise from system design decisions
  + Operational Risk Analysis
    - Concerned with the system user interface and risks from operator errors
* These phases are necessary because it is impossible to make all dependability and security decisions without complete info about the system implementation. Dependability and security requirements are particularly affected by technology choices and design decisions. Security requirements may have to be modified because they conflict with the security features that are provided by an off-the-shelf system.

**Safety Specification (Section 12.2)**

* Safety-critical systems are systems in which failures may affect the environment of the system and cause injury or death to the people in the environment.
* Goal: To identify requirements that will minimize the probability that such system failure occur
* Safety requirements are primarily protection requirements and are not concerned with normal system operation.
* In deriving these requirements you need to find a balance between safety and functionality and avoid overprotection
* Safety specification is usually focused on the hazards that may arise in a given situation, and the events that can lead to these hazards (Hazard = something that can result in death or injury)
* Process
  + Risk Identification
    - Hazard identification process that identifies hazards that may threaten the system
  + Risk Analysis
    - Process of hazard assessment to decide which hazards are the most dangerous and/or most likely to occur. These should be prioritized when deriving safety requirements
  + Risk Decomposition
    - Discovering events that can lead to the occurrence of a hazard.
    - AKA Hazard Analysis
  + Risk Reduction
    - Identification of safety requirements
    - Concerned with ensuring the hazard does not arise or lead to an accident or that if an accident does occur, the associated damage is minimized
* **Hazard Identification (Section 12.2.1)**
  + Different Classes of hazards such as physical, electrical, biological, radiation, service failure, and so on.
    - Each of these classes of hazards can then be analyzed to discover specific hazards that could occur
  + Experienced engineers, working with domain experts and professional safety advisers, identify hazards from previous experience and from an analysis of the application domain.
* **Hazard Assessment (Section 12.2.2)**
  + This stage focuses on understanding the probability that a hazard will occur and the consequences if an accident or incident associated with that hazard should occur
    - You need to make this analysis to understand whether a hazard is a serious threat to the system or to the environment
    - Analysis also provides a basis for deciding on how to manage the risk associated with the hazard
  + For each hazard, a statement of acceptability is created
    - This is expressed in terms of risk, where risk takes into account the likelihood of an accident and its consequences
    - 3 Risk Categories
      * Intolerable Risks (High costs for company)
        + Risks that threaten human life
        + System must be designed so that such hazards either cannot arise or, that if they do, features in the system will ensure that they are detected before they cause an accident
      * As Low As Reasonably Practical (ALARP) Risks
        + Risks that haves less serious consequences OR that are serious but have very low probability of occurring
        + System should be designed so that the probability of an accident arising because of a hazard is minimized, subject to other considerations such as cost and delivery
      * Acceptable Risks (Low costs for company)
        + Risks whose associated accidents normally result in minor damage
        + System designers should take all possible steps to reduce ‘acceptable’ risks, so long as these do not increase costs, delivery time, or other non-functional system attributes
* **Hazard Analysis (Section 12.2.3)**
  + Is the process of discovering the root causes of hazards in a safety-critical system
    - Find out what events or combination of events could cause a system failure that results in a hazard
  + 2 Approaches to finding causes
    - Top-down (Deductive)
      * Tend to be easier
      * Start with the hazard and work from that to the possible system failure
    - Bottom-up (Inductive)
      * Start with a proposed system failure and identify what hazards might result from that failure
  + Fault Tree analysis is an approach to hazard analysis
    - Here, you start with the hazards that have been identified
    - For each hazard, you then work backwards to discover the possible causes of that hazard
    - You put the hazard at the root of the tree and identify the system states that can lead to that hazard
    - For each of these states, you then identify further system states that can lead to them
    - You continue this decomposition until you reach the root causes of the risk
    - Hazards that only arise from a combination of root causes are usually less likely to lead to an accident than hazards with a single root cause
* **Risk Reduction (Section 12.2.4)**
  + Once potential risks and their root causes have been identified, you are then able to derive safety requirements that manage and ensure that incidents or accidents do not occur.
  + 3 Strategies
    - Hazard Avoidance
      * System is designed so that the hazard cannot occur
    - Hazard Detection and Removal
      * System is designed so that hazards are detected and removed before they result in an accident
    - Damage Limitation
      * System is designed so that the consequences of accident are minimized
  + Intolerable hazards may be handled by minimizing their probability and adding a protection system that provides a safety backup

**Reliability Specification (Section 12.3)**

* Is a measurable system attribute
* 2 Types of Reliability Requirements
  + Non-functional Requirements
    - Define the number of failures that are acceptable during normal use of the system, or the time in which the system is unavailable for use.
    - These are quantitative reliability requirements
  + Function Requirements
    - Define system and software functions that avoid, detect, or tolerate faults in the software and so ensure that these faults do not lead to system failure
* To achieve some required level of reliability, the function and design requirements of the system should specify the faults to be detected and the action that should be taken to ensure that these faults do not lead to system failures
* Process
  + Risk Identification
    - Identify the types of system failures that may lead to economic losses of some kind
  + Risk Analysis
    - Involves estimating the costs and consequences of different types of software failure and selecting high-consequence failures for farther analysis
  + Risk Decomposition
    - Do a root cause analysis of serious and probable system failures
  + Risk Reduction
    - Should generate quantitative reliability specifications that set out the acceptable probabilities of the different types of failures.
    - These should take into account the costs of failures. You may use different probabilities for different system services. You may also generate function reliability requirements
* Types of System Failure
  + Loss of Services
  + Incorrect Service Delivery
  + System/Data Corruption
* **Reliability Metrics (Section 12.3.1)**
  + Reliability can be specified as a probability that a system failure will occur when a system is in use
  + Metrics
    - Probability of Failure On Demand (POFOD)
      * You define the probability that a demand for service from a system will result in a system failure
      * This should be used in situations where a failure on demand can lead to a serious system failure
    - Rate of Occurrence of Failures (ROCOF)
      * Sets out the probable number of system failures that are likely to be observed relative to a certain time period (e.g. hour) or to the number of system executions.
      * The reciprocal of this is the Mean Time to Failure (MTTF)
        + Average number of time units between observed system failures
      * This should be used in situations where demands on systems are made regularly rather than intermittently
    - Availability (AVAIL)
      * Reflects a systems ability to deliver services when requested
      * Is the probability that a system will be operational when a demand is made for service
      * Also depends on the time required to get the system back into operation
      * Ex. AVAIL of 0.9999 mean that, on average, the system will be available for 99.99% of the operating time
  + To access the reliability of a system, you must capture the following data:
    - The number of system failures given a number of requests for system services (used for POFOD)
    - The time or number of transactions between system failures plus the total elapsed time or total number of transactions (used for ROCOC and MTTF)
    - The repair or restart time after a system failure that leads to loss of service (Use for Availability)
  + Time units that can be used include calendar, processor, or discrete (e.g., number of transactions) time
* **Non-Functional Reliability Requirements (Section 12.3.2)**
  + These requirements are quantitative specifications of the required reliability and availability of a system, calculated using one of the metrics described in the section above (12.3.1)
  + Advantages in deriving quantitative reliability specifications
    - Process of deciding what required level of the reliability helps to clarify what stakeholders really need. It helps stakeholders understand that there are different types of system failure, and it makes clear to them that high levels of reliability are very expensive to achieve
    - It provides a basis for assessing when to step testing a system
    - Is a means of assessing different design strategies intended to improve the reliability of a system
    - If a regulator has to approve a system before it goes into service, then evidence that a required reliability targets has been met is important for system certification
  + To establish the required level of system reliability, you have to consider the associate losses that could result from a system failure (can be financial losses, business reputations loss, etc.)
  + Difficulty arises with metrics because it is possible to overspecify reliability and thus incur higher development and validation costs. The reason for this is that system stakeholders find it difficult to translate their practical experience into quantitative specifications
  + Overspecification of reliability leads to very high costs
  + A fundamental problem with overspecification is that it may be practically impossible to show that a very high level of reliability or availability has been achieved
  + Organizations must therefore be realistic about whether it is worth specifying and validating a very high level of reliability. High reliability levels are clearly justified in system where reliable operation is critical
  + Steps to avoid overspecification
    - Specify the availability and reliability requirements for different types of failures. There should be a lower probability of serious failures occurring than minor failures
    - Specify the availability and reliability requirements for different services separately. Failures that affect the most critical services should be specified as les probable than those with only local effects
    - Decide whether you really need high reliability in a software system or whether the overall system dependability goals can be achieved in other ways
* **Function Reliability Specification (Section 12.3.3)**
  + This involves identifying requirements that define constraints and features that contribute to system reliability.
  + 3 Types of Function Reliability Requirements
    - Checking Requirements
      * These identify checks on inputs to the system to ensure that incorrect or out-of-range inputs are detected before they are processed by the system
    - Recovery Requirements
      * These requirements are geared to helping the system recover after a failure has occurred
      * Typically, these requirements are concerned with maintaining copies of the system and its data and specifying how to restore system services after a failure
    - Redundancy Requirements
      * Specify redundant features of the system that ensure that single component failure does not lead to a complete loss of service
  + These requirement may also include Process Requirements
    - These are requirements to ensure that good practice, known to reduce the number of faults in a system, is used in the development process.

**Security Specification (Section 12.4)**

* It is impractical to specify the specification of security requirements quantitatively
  + These are also usually “shall not” requirements
* Security is more challenging than safety because:
  + When considering safety, you can assume that the environment in which the system is installed is not hostile (nothing is trying to perform an attack)
  + When system failures occur that pose a risk to safety, you look for the errors or omissions that have caused the failure. When deliberate attacks cause system failures, finding the root cause may be more difficult as the attacker may try to conceal the cause of the failure
  + It is usually acceptable to shut down a system or to degrade system services to avoid a safety-related failure. However, attacks on a system may be so-called denial of service attacks. Shutting down that system means that the attack was successful.
  + Safety-related events are not generated by an intelligent adversary.
* Security requirements have to be more extensive than safety requirements
* 10 Types of Security Requirements that may be include in System Specification
  + Identification requirements specify whether or not a system should identify its users before interacting with them
  + Authentication requirements specify how users are identified
  + Authorization requirements specify the privileges and access permissions of identified users
  + Immunity Requirements specify how the system should protect itself against viruses, worms, and similar threats.
  + Integrity requirements specify how data corruption can be avoided
  + Intrusion Detection requirements specify what mechanisms should be used to detect attacks on the system
  + Non-repudiation requirements specify that a party in a transaction cannot deny its involvement in that transaction
  + Privacy requirements specify how data privacy is to be maintained
  + Security Auditing requirements specify how system use can be audited and checked
  + System Maintenance Security requirements specify how an application can prevent authorized changes from accidently defeating its security mechanisms
* As discussed in section 12.1, Risk Analysis and Assessment has 3 stages
  + Preliminary Risk Analysis
    - Decision on the detailed system requirements, the system design, or the implementation technology have not been made
    - The aim of this assessment process is to derive security requirements for the system as a whole
  + Life-Cycle Risk Analysis
    - This assessment takes place during the system development life cycle after design choices have been made
    - Additional security requirements take account of the technologies used in building the system and system design and implementation decisions
  + Operational Risk Analysis
    - This risk assessment considers the risks posed by malicious attacks on the operational system by users, with or without insider knowledge of the system
* Process Stages
  + Asset Identification
    - System assets that may require protection are identified. (Part of Risk Identification)
  + Asset Value Assessment
    - Where you estimate the value of the identified assets (Part of Risk Analysis)
  + Exposure Assessment
    - Where you assess the potential losses associated with each asset (Part of Risk Analysis)
  + Threat Identification
    - Where you identify the threats to system assets (Part of Risk Analysis)
  + Attack Assessment
    - Where you decompose each threat into attacks that might be made on the system and the possible ways in which these attacks may occur (Part of Risk Decomposition)
  + Control Identification
    - Where you propose the controls that might be put in place to protect an asset. (Part of Risk Reduction)
  + Feasibility Assessment
    - Where you assess the technical feasibility and the costs of the proposed controls. (Part of Risk Reduction)
  + Security Requirements Definition
    - Where knowledge of the exposure, threats, and control assessments is used to derive system security requirements. (Part of Risk Reduction)

Formal Specification (Section 12.5)

* Formal methods are mathematically-based approaches to software development where you define a formal model of the software
* To create a model, you translate system user requirements, which are expressed in natural language, diagrams, and tables, into a mathematical language which has formally defined semantics
* Formal specifications are essential for a verification of the design and implementation of software. They are also the most precise way of specifying systems, and so reduce the scope for misunderstanding.
* These specifications are usually developed as part of a plan-based SW process, where the system is completely specified before development.
* Formal Specification in a Plan-based SW Process
  + User Requirements Definition
  + System Requirements Specification
  + Architectural Design
  + Formal Specification
  + High-Level Design
* Advantages of developing a formal specification are:
  + As you develop a formal specification in detail, you develop a deep and detailed understanding of the system requirements
  + As the specification is expressed in a language with formally define semantics, you can analyze it automatically to discover inconsistencies and incompleteness
  + If you use a methods such as the B method, you can transform the formal specification into a program through a sequence of correctness-preserving transformations
  + Program testing costs may be reduced because you have verified the program against its specification
* Arguments against using Formal Specification
  + Problem owners and domain experts cannot understand a formal specification so they cannot check that it accurately represent their requirements
  + It is fairly easy to quantify the costs of creating a formal specification, but more difficult to estimate the possible cost savings that will result from its use
  + Most SW engineers have not been trained to use formal specification languages
  + It is difficult to scale current approaches to formal specification up to very large system
  + Formal specification is not compatible with agile methods of development