CSE- 321
Software Engineering

Lecture : 05

Systems Engineering

Fahad Ahmed

Lecturer, Dept. of CSE

E-mail: fahadahmed@uap-bd.edu

Lecture Outlines

- Green Software Engineering
- Sustainable Software Engineering
- Case Study
- Systems Engineering
- Legacy Systems
- Critical System
- System Dependability
- Software Development Lifecycle
- Software Requirement Specification

Environmental Ethics and Engineering

Environmental Ethics

Deals with relationship of man with environment.

Some common topics in environmental ethics are

- Co-create rules and regulations for the ethical use of technology
- **green technology**
- Electric vehicles
- Wave energy
- **Green computing**
- Wind power
- Wind turbine
- Solar power
- Ocean thermal energy conversion

Green Software Engineering

What is a green software?

"**Computer software that can be developed and used efficiently and effectively with minimal or no impact to the environment**".

Green Software Engineering is an emerging discipline at the intersection of climate science, software practices and architecture, electricity markets, hardware and data center design.

The **Principles of Green Software Engineering** are a core set of competencies needed to define, build and run green sustainable software applications.

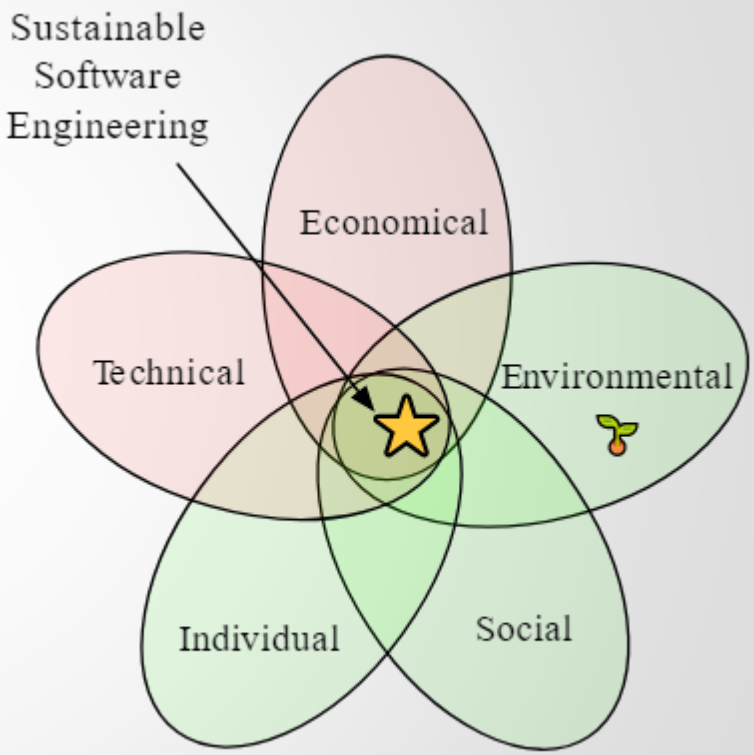
Green Software Engineering >> Sustainable Software Engineering

1. Electricity: Build applications that are energy efficient.
2. Carbon Intensity: Consume electricity with the lowest carbon intensity.
3. Embodied Carbon: Build applications that are hardware efficient.
4. Energy Proportionality: Maximize the energy efficiency of hardware.
5. Networking: Reduce the amount of data and distance it must travel across the network.
6. Demand Shaping: Build carbon-aware applications.
7. Measurement & Optimization: Focus on step-by-step optimizations that increase the overall carbon efficiency.

Sustainable Software Engineering

Sustainable software development is a comprehensive approach that centers around two main ideas. First, product teams should make their solutions as cost-efficient, productive, and eco-friendly as possible. Second, optimization should increase the quality of products.

By definition, sustainability covers five main perspectives:
environmental, **social**, **individual**, **economic, technical.**



Sustainable Software Engineering

Software Engineering (SE) has long addressed sustainability by **narrowing it down to economic and technical sustainability**.

However, our society is facing major sustainability challenges that can no longer be overlooked by software engineers and computer scientists. It was estimated that, **by 2040, the ICT sector will contribute to 14% of the global carbon footprint**.

Hence, environmental, social, and individual ought to be part of the equation when it comes to design, build, and release software systems. The problem is far from simple, but **we need expert computer scientists** to bring sustainability into the core values of the next generation of tech-leading organizations.

Sustainable Software Engineering

•For example, **E-bay** helps to reuse objects, instead of putting them to waste, and **Google Maps** shows public transport routes, making it easier to act in an environmentally friendly way.

Case #1

Following Time, a few years ago, Google was actively building its data centers throughout the US, including the arid regions of Arizona. Meanwhile, the company withheld information about water consumption for cooling centers, citing *commercial secrecy*. But the truth comes out. In 2019, the enterprise ordered 2.3 billion gallons of water for this purpose.

According to Water Smart, this volume is enough to irrigate over 15,375 golf courses or keep 20,000 households!

Link: https://www.jevera.software/post/sustainable-software-products-development-how-to-follow-the-new-eco-trend

Case #2

One more digital giant, Microsoft, also distinguished itself by wasteful water consumption for cooling the same data centers. And here's what they came up with. To reduce water consumption, the company decided to build data centers on the seabed. In 2018, they deployed a Northern Isles datacenter near **Scotland**. It contains 864 servers and boasts a sustainable cooling system infrastructure.

Link: https://www.jevera.software/post/sustainable-software-products-development-how-to-follow-the-new-eco-trend

How AI technology lead to a more sustainable future??

Case-Study:

One of the biggest retail corporations represents multiple deployments of digital transformations that work to eliminate wastage and energy usage and to provide supply chain control.

First of all, numerous **built-in IoT sensors and shelf-scanning robots** prove to be sustainable in terms of energy savings and customer experience.

How AI technology lead to a more sustainable future??

Case-Study: Walmart

Also, Walmart is a successful e-retailer that provides efficient online services, like Mobile Express Returns and QR code scanning. It enables their customers to shop staying at home thus diminishes transport usage and CO2 emissions.

Walmart is constantly developing innovative ideas that can be implemented not only within the retail branch. In 2018 the corporation patented the idea of a robobee – a self-manned drone for pollinating crops equipped with cameras and sensors. This tool also makes it possible to **detect agricultural problems** and get more sufficient control over the Walmart food supply chain that, consequently, minimizes food waste.

Tech-enabled innovations can bring us not only business profits but also a great social and environmental impact.

System

What is a system?

- A purposeful **collection of inter-related components** working together to **achieve some common objective**.
- A system may include software, mechanical, electrical and electronic hardware and be operated by people.
- System components are dependent on other system components.

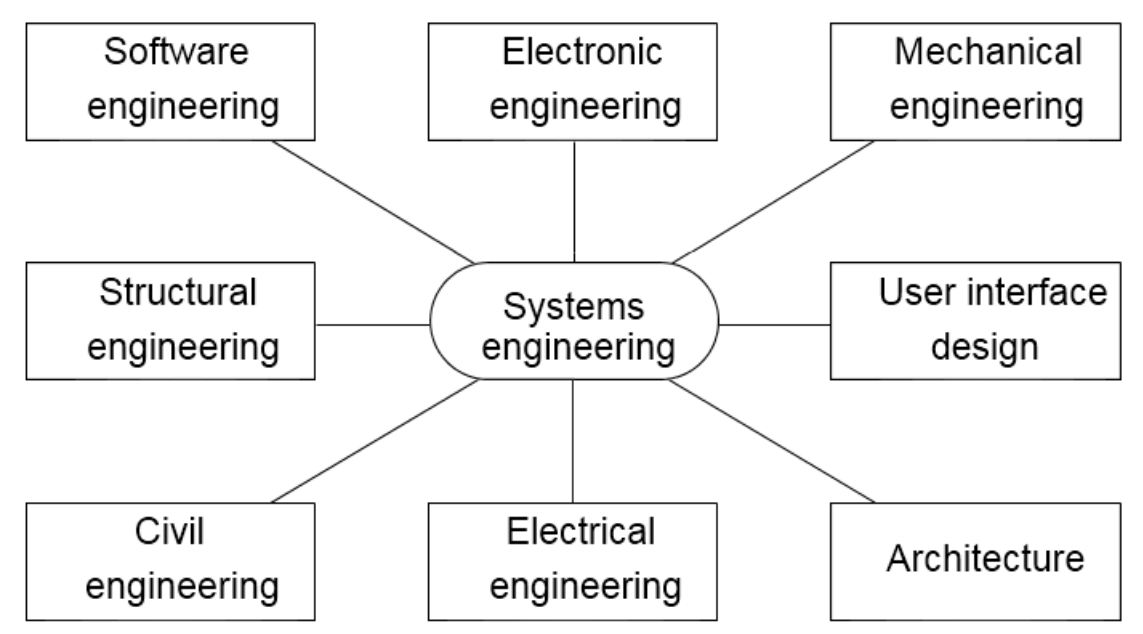
- **System categories**
  - Technical computer-based systems
  - Socio-technical systems

Systems engineering

Systems engineering is the activity of specifying, designing, implementing, validating, deploying and maintaining socio-technical systems.

Systems engineers are not just concerned with software but also with hardware and the system's interactions with users and its environment.

Concerned with the services provided by the system, constraints on its construction and operation and the ways in which it is used.



Disciplines involved in systems engineering

Legacy systems

Legacy systems

Legacy systems is an old or out-dated system, technology or software application that continues to be used by an organization because it still performs the functions it was initially intended to do.

– Bank customer accounting system.

Legacy software is software that has been around a long time and still fulfills a business need. It is mission critical and tied to a particular version of an operating system or hardware model (vendor lock-in) that has gone end-of-life. Generally the lifespan of the hardware is shorter than that of the software.

Legacy systems

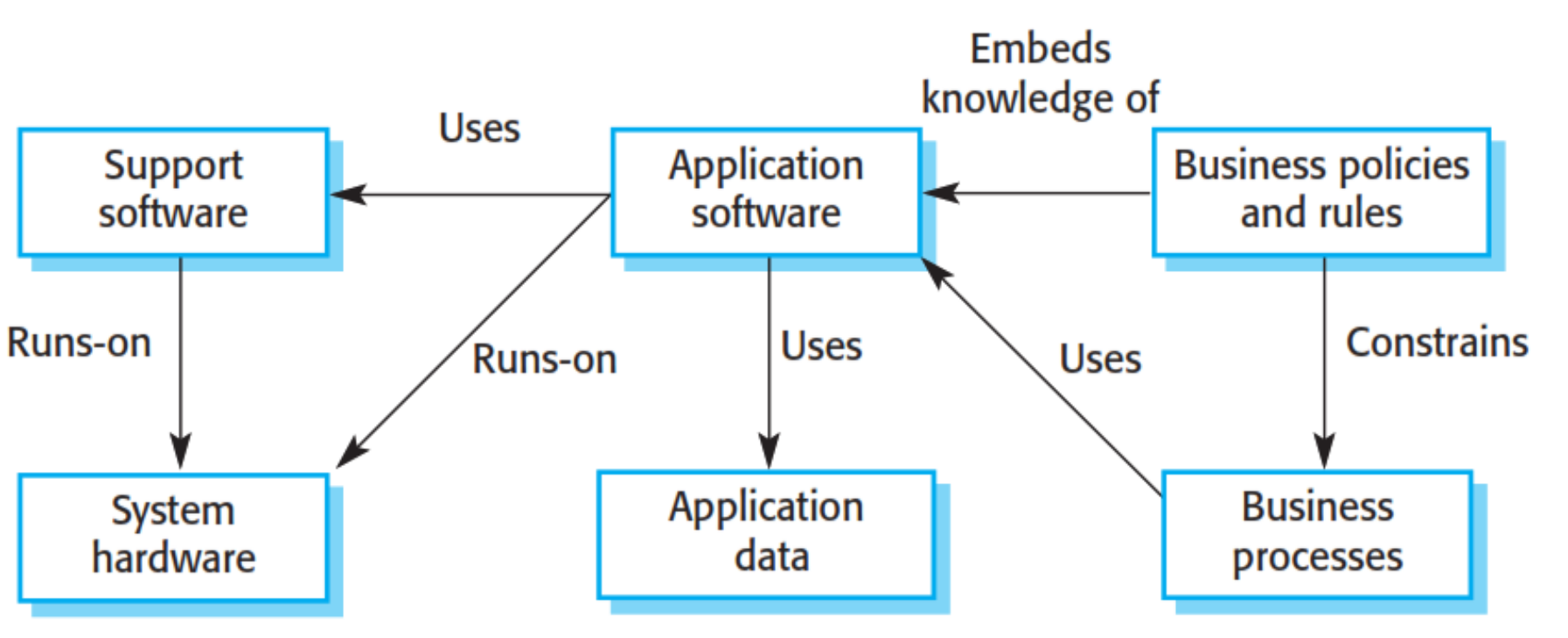
Examples of Legacy Software

Microsoft Windows 7: Windows 7 officially became a legacy operating system in January 2020 after Microsoft halted security updates and support for it. However, over 100 million machines continue to run this operating system.

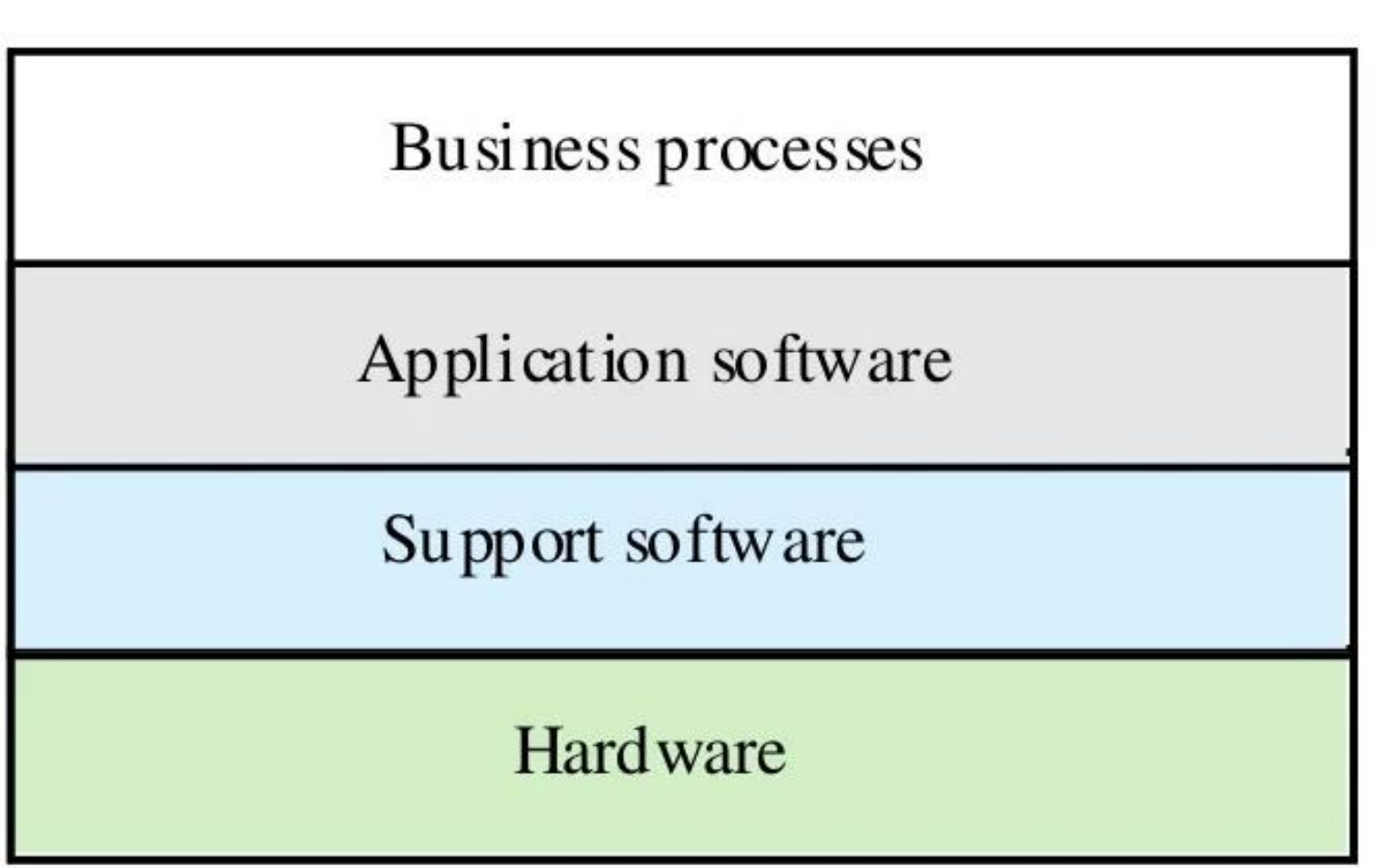
COBOL: Common Business-Oriented Language or COBOL is still used 55 years after its development. Forty-eight percent of businesses and government organizations reportedly depend on this language more than others.

Discontinued Oracle products: Oracle database software such as E-Business Suite and Peoplesoft.

Legacy system components



Legacy system replacement: Layered model



Legacy system

What are some of the most challenging aspects of changing a legacy system for a new one?

The replacement and migration of legacy systems is high in the list of critical elements within organizations. Replacing a legacy system is challenging, costly and time consuming. Because of :

1. **Costs**
2. **Technical specifications**
3. **Data protection**
4. **User experience**

Top Legacy System Modernization Trends

Aside from migrating to the cloud, important tools to modernize legacy systems and increase an organization's operational efficiency include:

Automation

Legacy systems often lack automation in many areas. By automating repetitive tasks, human errors and operational costs can be eliminated.

DevOps

DevOps consultants and engineers can assist in selecting the most efficient services to use. The DevOps approach to programming results in increased ROI with faster delivery times.

Container-Based Applications

Using containers in the migration process allows applications to be easily moved between infrastructures with minimal code changes.

Microservice Architecture

This approach uses separate codebases for each service. It enables the deployment of a service without the expense of rebuilding the entire application.

Critical System

It is the systems that fail to deliver the expected objective in which any system failure can result in economic losses, physical damage or human life loss.

There are three main types of critical systems:

- **Safety-critical systems**
  - **Failure** results in loss of life, damage to the environment;
  - **Chemical plant protection** system
- **Mission-critical systems**
  - **Failure** results in failure of some goal-directed activity;
  - **Spacecraft navigation** system
- **Business-critical systems**
  - **Failure** results in high economic losses;
  - Customer **accounting system** in a bank

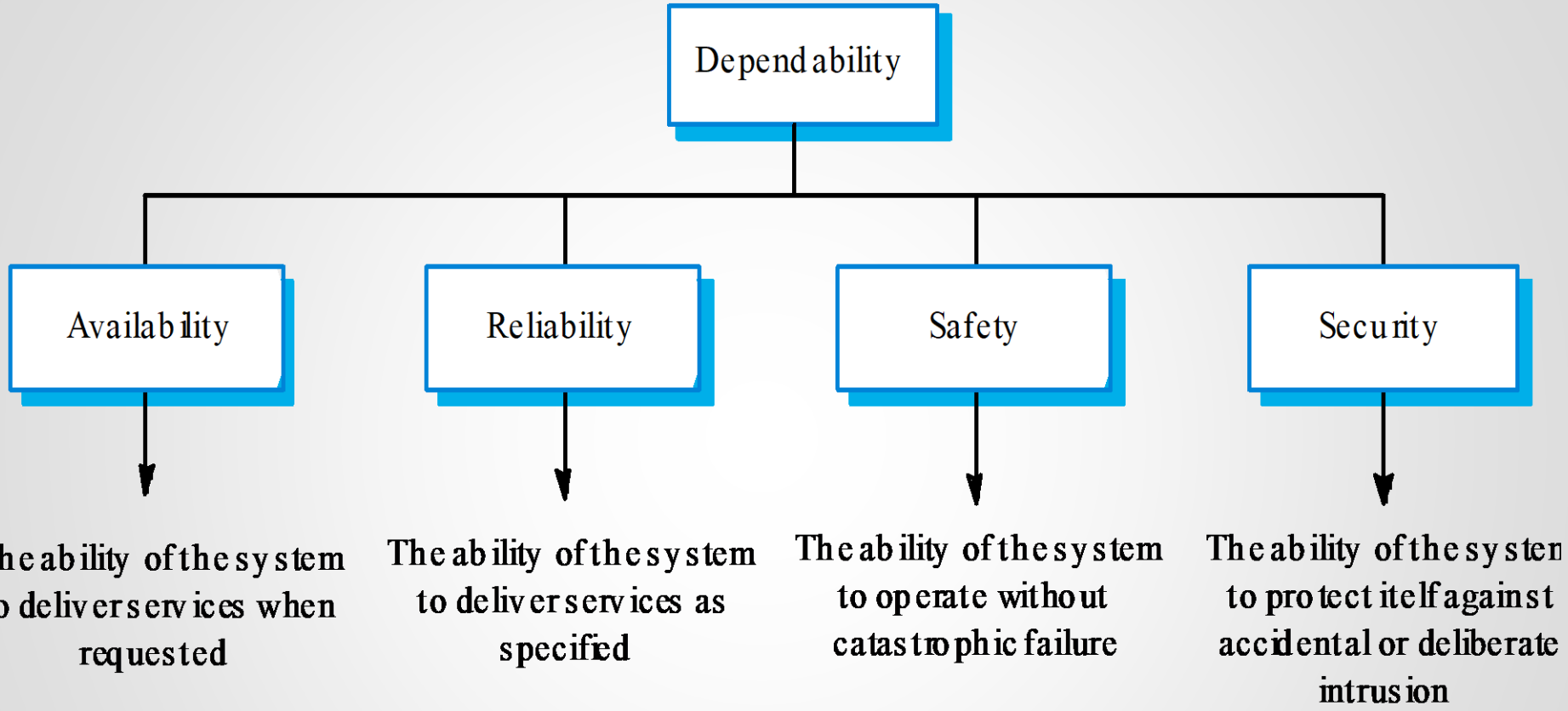
System Dependability

The dependability of a system **reflects the user's degree of trust in that system**. It reflects the extent of the user's confidence that it will operate as users expect and that it will not 'fail' in normal use.

Principal **dimensions of dependability** are:

- Availability
- Reliability
- Safety
- Security

Dimensions of System Dependability



Other dependability properties

- **Repairability**
  Reflects the extent to which the system can be repaired in the event of a failure.

- **Maintainability**
  Reflects the extent to which the system can be **adapted to new requirements.**

- **Survivability**
  Reflects the extent to which the system can deliver services whilst under hostile attack.

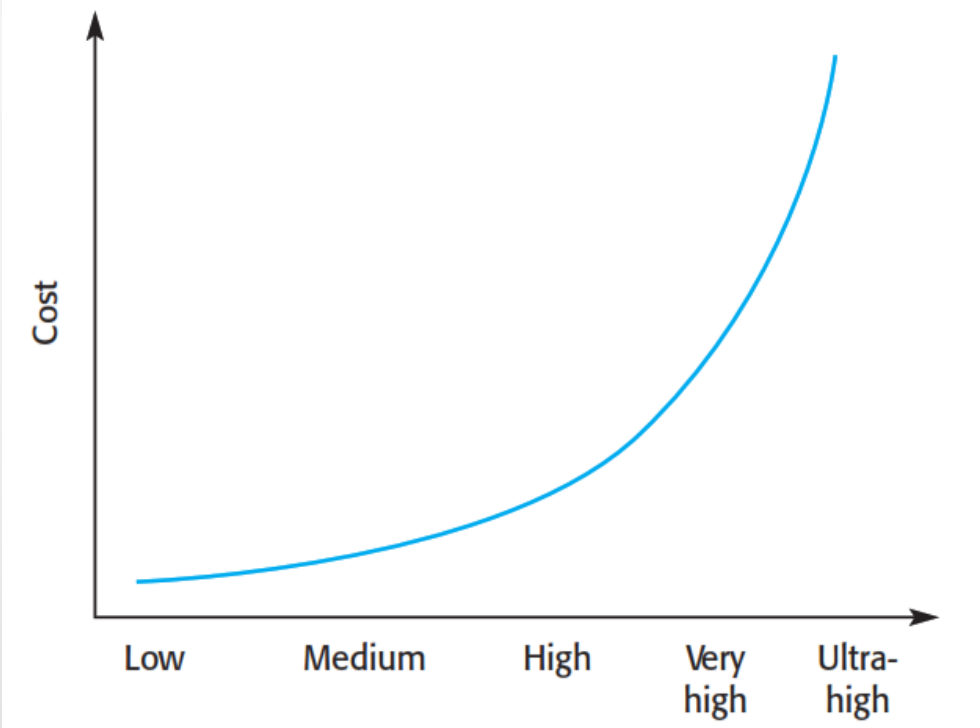
- **Error tolerance**
  Reflects the extent to which user input errors can be avoided and tolerated.

Dependability Costs Curve

Dependability costs tend to increase **exponentially** as **increasing levels of dependability** are required

There are two reasons for this

- The use of **more expensive development techniques and hardware** that are required to achieve the **higher levels of dependability**
- The increased testing and system validation that is required to **convince the system client** that the required levels of dependability have been achieved

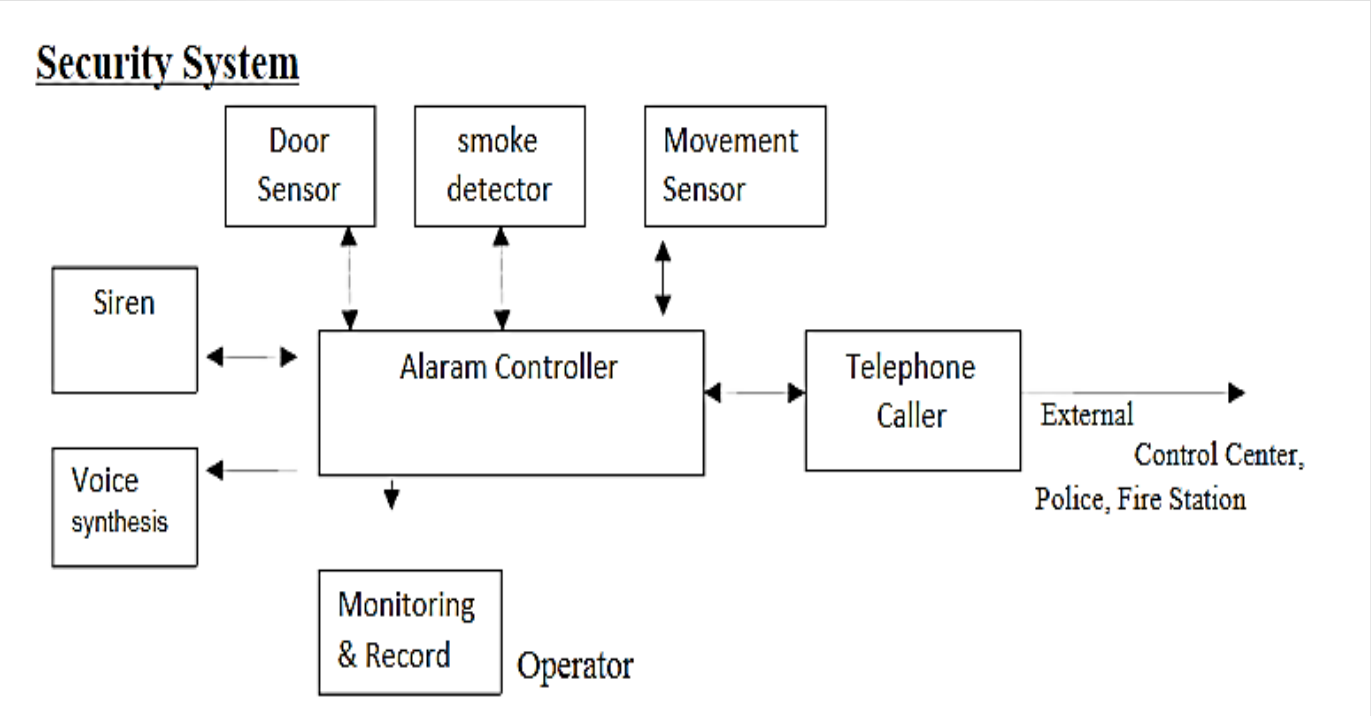


Sample Question

Consider a security system which is intended to protect against intrusion and to detect fire. It incorporates **smoke sensors, movement sensors, door sensors, video cameras** under computer control, **located at various places** in the building, an operator console where the system status is reported, and external communication facilities to call the appropriate services such as the police and fire departments.

Draw a block diagram of a possible design for such a system.

Sample Question



Security System

Door sensor - To monitor who enter the building and room

Movement Sensor - To monitor and alert if unauthorize person is enter.

Smoke detector - To detect smoke, typically as an indicator of fire.

Monitoring and Record sub system - To monitor building and report to operator. To record all CCTV camera.

Telephone caller - To submit external control center, township police statation, Township fire station.

Sample Question

Does safety always increase by increasing system reliability?

No, safety does not necessarily change at all by increasing system reliability. Safety is another dimension.

Software Development Life Cycle

What is SDLC?

SDLC stands for **Software Development Lifecycle**. **SDLC** is a systematic process for building software that ensures the quality and correctness of the software built. SDLC process aims to produce high-quality software that meets customer expectations.

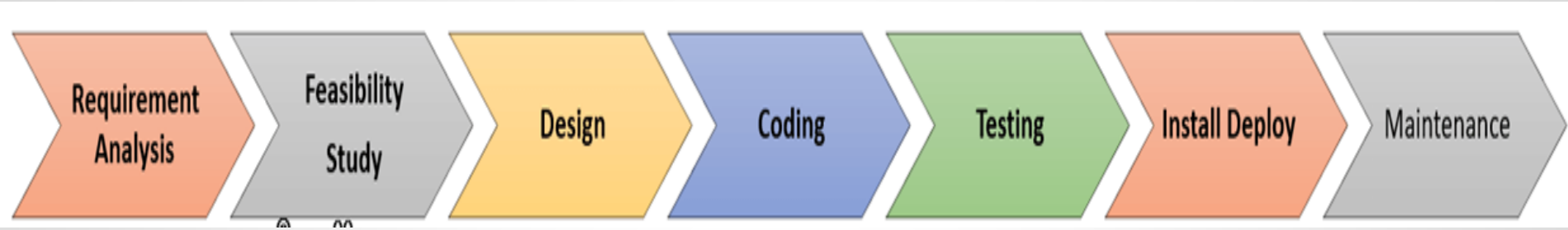
Why SDLC?

- It offers a basis for project planning, scheduling, and estimating
- Provides a framework for a standard set of activities and deliverables
- It is a mechanism for project tracking and control
- Increases visibility of project planning to all involved stakeholders of the development process
- Increased and enhance development speed
- Improved client relations
- Helps you to decrease project risk and project management plan overhead

Software Development Life Cycle

SDLC Phases

The entire SDLC process divided into the following stages:



Phase 1: Requirement collection and analysis

Phase 2: Feasibility study

Phase 3: Design

Phase 4: Coding

Phase 5: Testing

Phase 6: Installation/Deployment

Phase 7: Maintenance

Software Development Life Cycle

Phase 2: Feasibility study

This process conducted with the help of '**Software Requirement Specification**' document also known as '**SRS**' document.

There are mainly five types of feasibilities checks:

1. Economic
2. Legal
3. Operation feasibility
4. Technical
5. Schedule

Phase 3: Design:

In this third phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture.

Software Development Life Cycle

Phase 3: Design:

There are two kinds of design documents developed in this phase:

1. **High level design (HLD):** It give the architecture of software product.
2. **Low level design (LLD):** It describe how each and every feature in the product should work and every component.

High-Level Design (HLD)

- Brief description and name of each module
- An outline about the functionality of every module
- Interface relationship and dependencies between modules
- Database tables identified along with their key elements
- Complete architecture diagrams along with technology details

Software Development Life Cycle

Phase 3: Design:

Low-Level Design(LLD)

- Functional logic of the modules
- Database tables, which include type and size
- Complete detail of the interface
- Addresses all types of dependency issues
- Listing of error messages
- Complete input and outputs for every module

Software Development Life Cycle

Phase 7: Maintenance:

Once the system is deployed, and customers start using the developed system, following 3 activities occur

1. **Bug fixing** - bugs are reported because of some scenarios which are not tested at all
2. **Upgrade** - Upgrading the application to the newer versions of the Software
3. **Enhancement** - Adding some new features into the existing software

For more details: https://www.guru99.com/software-development-life-cycle-tutorial.html

Software Requirement Specification

Software requirement specification(SRS) is a kind of document which is created by a software analyst after the requirements collected from the various sources - the requirement received by the customer written in ordinary language.

It is the job of the analyst to **write the requirement in technical language** so that they can be understood and beneficial by the development team.

The models used at this stage include **ER diagrams**, **data flow diagrams (DFDs)**, function decomposition diagrams (FDDs), **data dictionaries**, etc.

Data Flow Diagrams:

Data Flow Diagrams (DFDs) are used widely **for modeling** the requirements. DFD shows the flow of data through a system. The system may be a company, an organization, a set of procedures, a computer hardware system, a software system, or any combination of the preceding. The DFD is also known as a **data flow graph or bubble chart.**

Data Dictionaries:

Data Dictionaries are simply repositories to **store information about all data items** defined in DFDs. At the requirements stage, the data dictionary should at least define customer data items, to ensure that the customer and developers use the same definition and terminologies.

Entity-Relationship Diagrams:

Another tool for requirement specification is the entity-relationship diagram, often called an "***E-R diagram***." It is a detailed **logical representation of the data** for the organization and uses three main constructs i.e. data entities, relationships, and their associated attributes.

Software Requirement Specification

Why Use an SRS Document?

An SRS gives you a complete picture of your entire project. It provides a single source of truth that every team involved in development will follow. It is your plan of action and keeps all your teams — from development to maintenance — on the same page.

Software Requirements Specification vs. System Requirements Specification

A **software requirements specification (SRS)** includes in-depth descriptions of the software that will be developed.

A **system requirements specification (SyRS)** collects information on the requirements for a system.

Software Requirement Specification

How to write a SRS Document?

1. **Introduction**
   - 1.1 Purpose
   - 1.2 Document conventions
   - 1.3 Project scope
   - 1.4 References
2. **Overall description**
   - 2.1 Product perspective
   - 2.2 User classes and characteristics
   - 2.3 Operating environment
   - 2.4 Design and implementation constraints
   - 2.5 Assumptions and dependencies
3. **System features**
   - 3.x System feature X
     - 3.x.1 Description
     - 3.x.2 Functional requirements
4. **Data requirements**
   - 4.1 Logical data model
   - 4.2 Data dictionary
   - 4.3 Reports
   - 4.4 Data acquisition, integrity, retention, and disposal

Software Requirement Specification

How to write a SRS Document?

5. External interface requirements
- 5.1 User interfaces
- 5.2 Software interfaces
- 5.3 Hardware interfaces
- 5.4 Communications interfaces

6. Quality attributes
- 6.1 Usability
- 6.2 Performance
- 6.3 Security
- 6.4 Safety
- 6.x [others]

7. Internationalization and localization requirements

8. Other requirements

Appendix A: Glossary

Appendix B: Analysis models

Software Requirement Specification

How to write a SRS Document?

1. Introduction

- 1.1 Purpose
- 1.2 Intended Audience
- 1.3 Intended Use
- 1.4 Scope
- 1.5 Definitions and Acronyms

2. Overall Description

- 2.1 User Needs
- 2.2 Assumptions and Dependencies

3. System Features and Requirements

- 3.1 Functional Requirements
- 3.2 External Interface Requirements
- 3.3 System Features
- 3.4 Nonfunctional Requirements

Thanks to All