HANDOUTS (MID TERM)

WEEK 1-8

Prepared By VURANK

Introduction to the Discipline of Design:

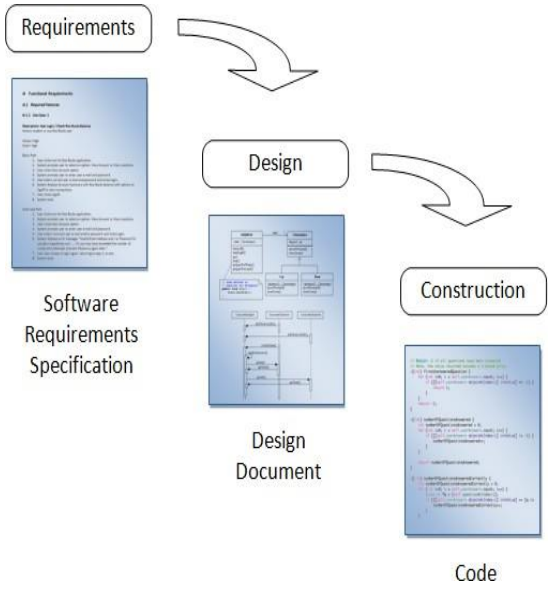
- We live in a designed world.
- Design is economically important and effects our quality of life
- Any product that is an aggregate of more primitive elements, can benefit from the activity of design.

Software Products:

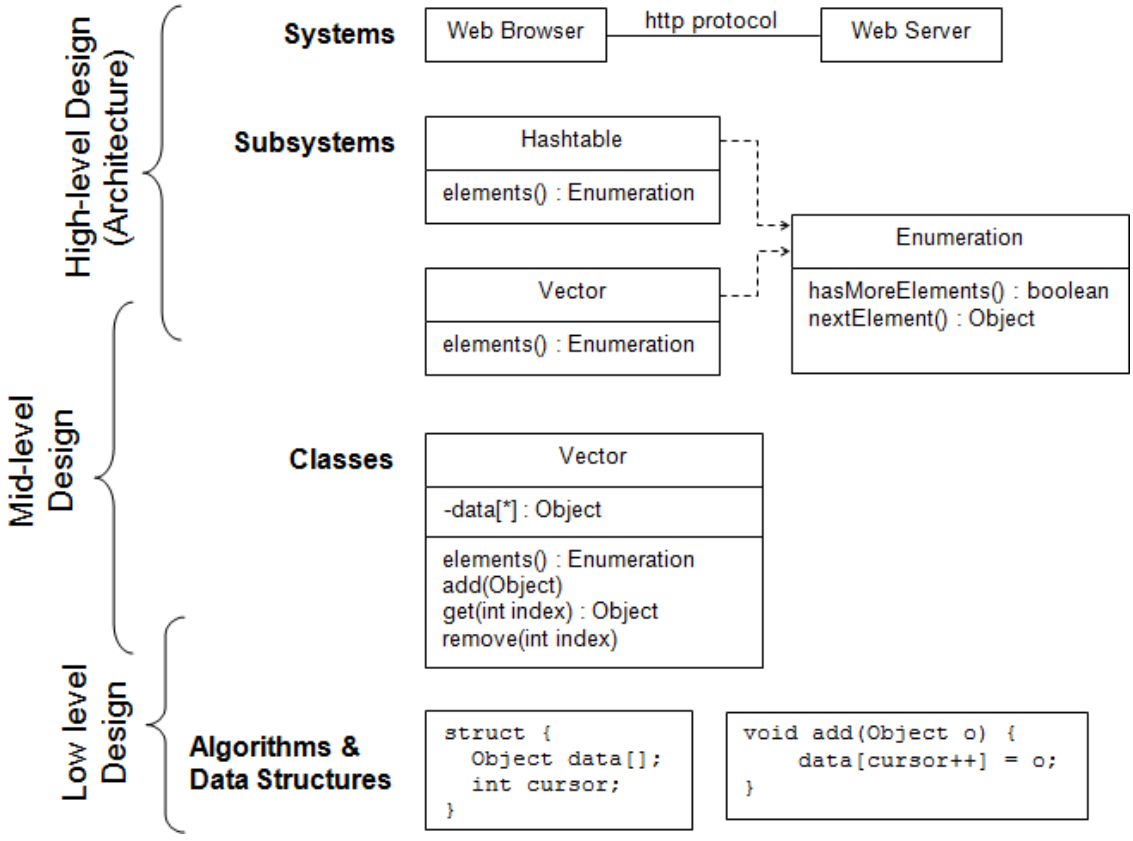
- **Software** is any executable entity, such as a program, or its parts, such as sub-programs.
- A software product is an entity comprised of one or more programs, data, and supporting materials and services that satisfies client needs and desires either as an independent artifact or as essential ingredient in some other artifact.

What Is Software Design?

- Software design is the activity of specifying the mature and composition of software products that satisfy client needs and desire, subject to constraints.
- Software designers do what designers in other disciplines do, except they do it for software products.
- Design bridges that gap between knowing what is needed (software requirements specification) to entering the code that makes it work (the construction phase).



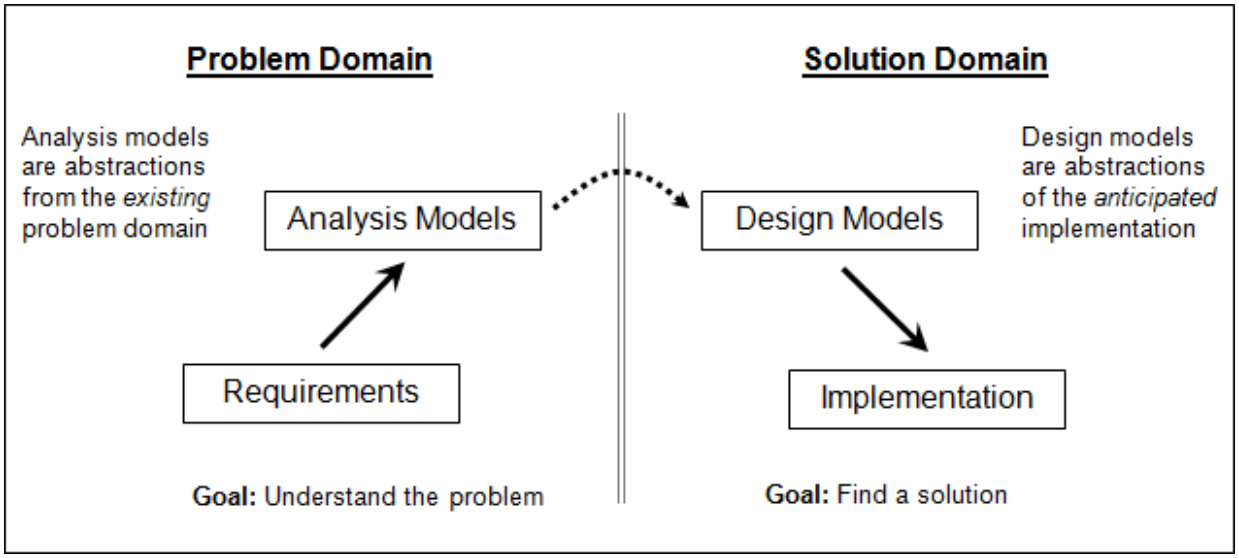
Design Occurs at Different Levels:



Standard Levels of Design

Why Design is Hard?

- Design is difficult because design is an abstraction of the solution which has yet to be created



Design as Problem Solving:

- An especially fruitful way to think about design is as problem solving.

Advantages:

1) Suggests partitioning information between problem and solution
2) Emphasizes that there may be more than one good solution (design)
3) Suggests techniques such as changing the problem, trial and error, brainstorming, etc.

Abstraction:

- Abstraction is an important problem-solving technique, especially in software design
- Abstraction is suppressing or ignoring some properties of objects, events, or situations in favor of others.

Importance of Abstraction:

1. **Problem simplification:**
   - Abstracting allows us to focus on the most important aspects of a problem in (partially) solving it.

2. **Structuring problem solving**
   - **Top-down strategy**: Solve an abstract version of the problem, then add details (refinement)
   - **Bottom-up strategy**: Solve parts of a problem and connect them for a complete solution

What is Model?

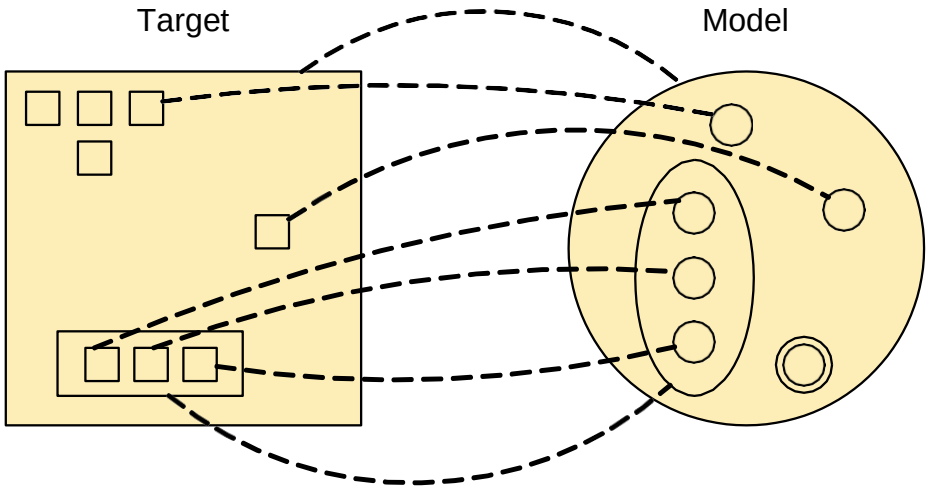
- A **model** is an entity used to represent another entity (the *target*) by establishing

a) A correspondence between the parts or elements of the target and the parts or elements of the model

b) A correspondence between relationships among the parts or elements of the target and relationships among the parts or elements of the model.

Modeling:

A model represents a target by having model parts corresponding to target parts, with relationships between model parts corresponding to relationships between target parts.



Modeling in Design:

- Modeling is used for the following purposes:
  1. Problem understanding
  2. Design creation and investigation
  3. Documentation
- Modeling work because models abstract details of the target.
- Models can fail if important and relevant details are left out.

Modeling in Software Design:

Software design models may be divided into two broad classes: static and dynamic models

1) A static model represents aspects of programs that do not change during program execution.
2) A dynamic model represents what happens during program execution.

Static and Dynamic Models:

- **Static model** examples include object and class models, component and deployment diagrams, and data structure diagrams.
- **Dynamic model** examples include use case descriptions, interaction diagrams, and state diagrams.

Introduction to the Discipline of Design:

Varieties of Design:

- **Product design** is a discipline that arose during the Industrial Revolution and is now an established field whose practitioners specify products.
- The major issues in product design are aesthetics, product features and capabilities, usability, manageability, manufacturability, and operability.

- **Engineering design** is the activity of specifying the technical mechanisms and workings of a product. Engineers apply mathematical and scientific principles and techniques to work out the technical details of complex products.
- Product designers and engineers often work together in design teams *to specify large and complex products.*

Product Designer vs. Engineering Designer

1) Product designers are concerned with styling and aesthetics, function and usability, manufacturability and manageability.
2) Industrial designers, (building) architects, interior designers, graphic designers, etc.
3) Engineering designers are concerned with technical mechanisms and workings.

4) Structural, mechanical, chemical, and electrical engineers

5) Product designers handle the "externals" of product design while engineers take care of the "internal" technical details.

Design Teams:

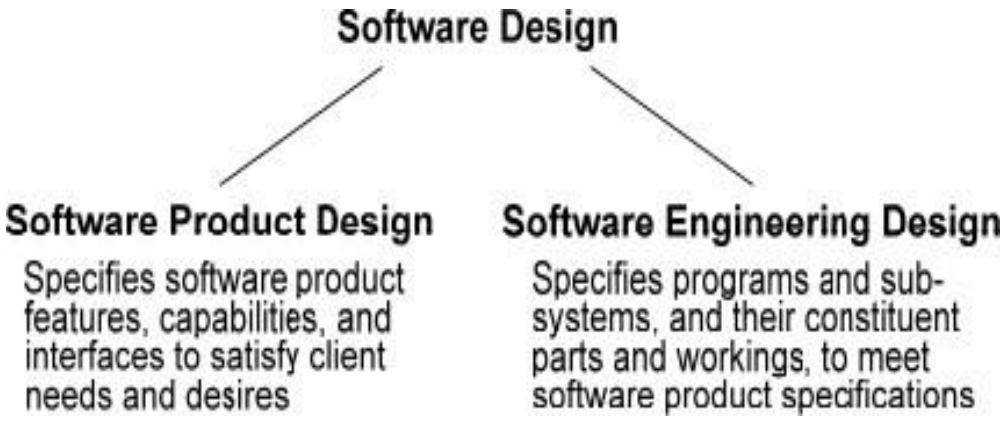
The talents and skills of both product designers and engineers are needed to design such things. Table 1 illustrates the complementary responsibilities of product and engineering designers for several products.

Product	Product Designers	Engineering Designers
Recliner	Size, styling, fabrics, and controls	Reclining mechanism and frame
Clothes Drier	Capacity, features (timed dry, permanent press cycle, etc.), dimensions, controls and how they work, styling, and colors	Frame, cylinder, drive and fan motors, heating elements, control hardware and software, electrical and mechanical connections, and materials
Clock Radio	Features (number of alarms, snooze alarm, etc.), displays, controls and how they work, case and control styling and colors	Clock, radio, display, digital and mechanical control and interface hardware, control software, electrical and mechanical connections, and materials
Refrigerator	Capacity, features (ice maker, ice-water spigot, etc.), dimensions, number and arrangement of compartments, shelves, doors, controls and how they work, colors, lighting, and styling	Refrigeration mechanisms, insulation, water storage, pumps, plumbing, electrical wiring and connections, mechanical frame and connectors, and materials

Table 1: Product and Engineering Designers' Responsibilities

Software design:

The field of software design can be divided into two sub-fields that each demand considerable skill and expertise: software product design and software engineering design.



Software Product Design:

- Software product design is the activity of specifying software product features, capabilities, and interfaces to satisfy client needs and desires.
- Requires skills in user interface and interaction design, communications, industrial design, and marketing

Software Engineering Design:

- Software engineering design is the activity of specifying programs and sub-systems, and their constituent parts and workings, to meet software product specifications.
- Requires skills in programming, algorithms, data structures, software design principles, practices, processes, techniques, architectures, and patterns

---

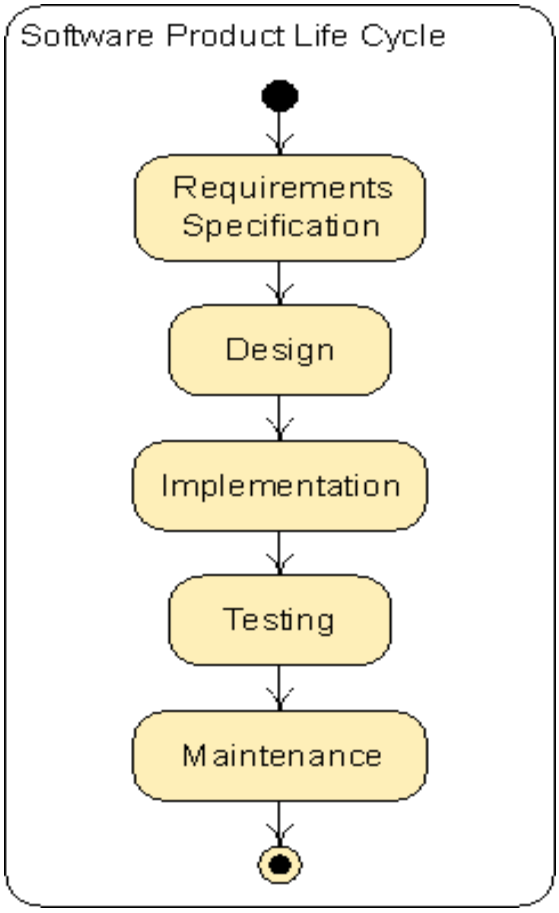
(WEEK 1 COMPLETED)

WEEK #2

Software Design in the Life Cycle:

The software life cycle is the sequence of activities through which a software product passes from initial conception through retirement from service.

Waterfall Life Cycle Model:



The waterfall model captures the logical, but not the temporal, relationships between software development activities.

Requirements Specification Activity:

1) The goal of the requirements specification activity is to specify a product satisfying the needs and desires of clients and other interested parties.
2) **Specifications are recorded in a software requirements specification (SRS).**
3) We assume that every SRS includes a user interface design.

4) Factors that limit the range of design solutions, such as cost, time, size, user capability, and required technology, are called design **constraints. Design** constraints are usually given as part of the problem specification.

Design Activity:

- During the design activity, developers figure out how to build the product specified in the SRS. This includes selecting an overall program structure, specifying major parts and sub-systems and their interactions, then determining how each part or sub-system will be built.
- The result of the design activity is a **design document** recording the entire design specification. The design document solves the (engineering) design problem posed in the SRS.

Implementation Activity:

- Code is written in accord with the specifications in the design document. The product of the implementation activity is a more or less finished, working program satisfying the SRS.
- Programming essentially includes some engineering design work.

Testing Activity:

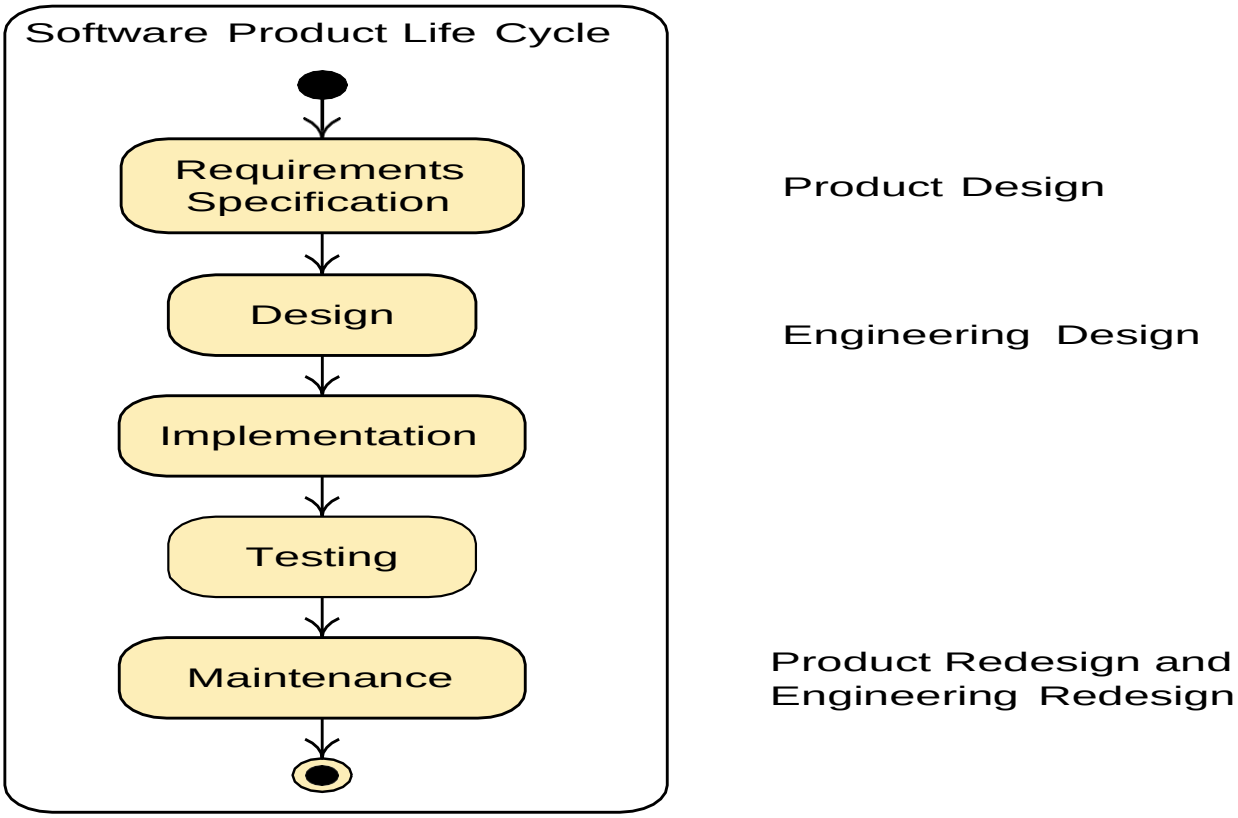
- Programs are run during the testing activity to find bugs.
- Testing is usually done bottom up, with small parts or program units tested alone, and then integrated collections of program units tested as separate sub-systems, and finally the entire program tested as a whole.

Maintenance Activity:

- Maintenance activity occurs after a product has been deployed to clients.
- After deployment to clients, products are corrected, ported, and enhanced during maintenance activities.
- Product design occurs during the requirements specification and maintenance activities, and engineering design occurs during the design, implementation, and maintenance activities.

Design across the Life Cycle:

Design across the Life Cycle Figure illustrates how software design activities are spread across the life cycle.



"What" Versus "How":

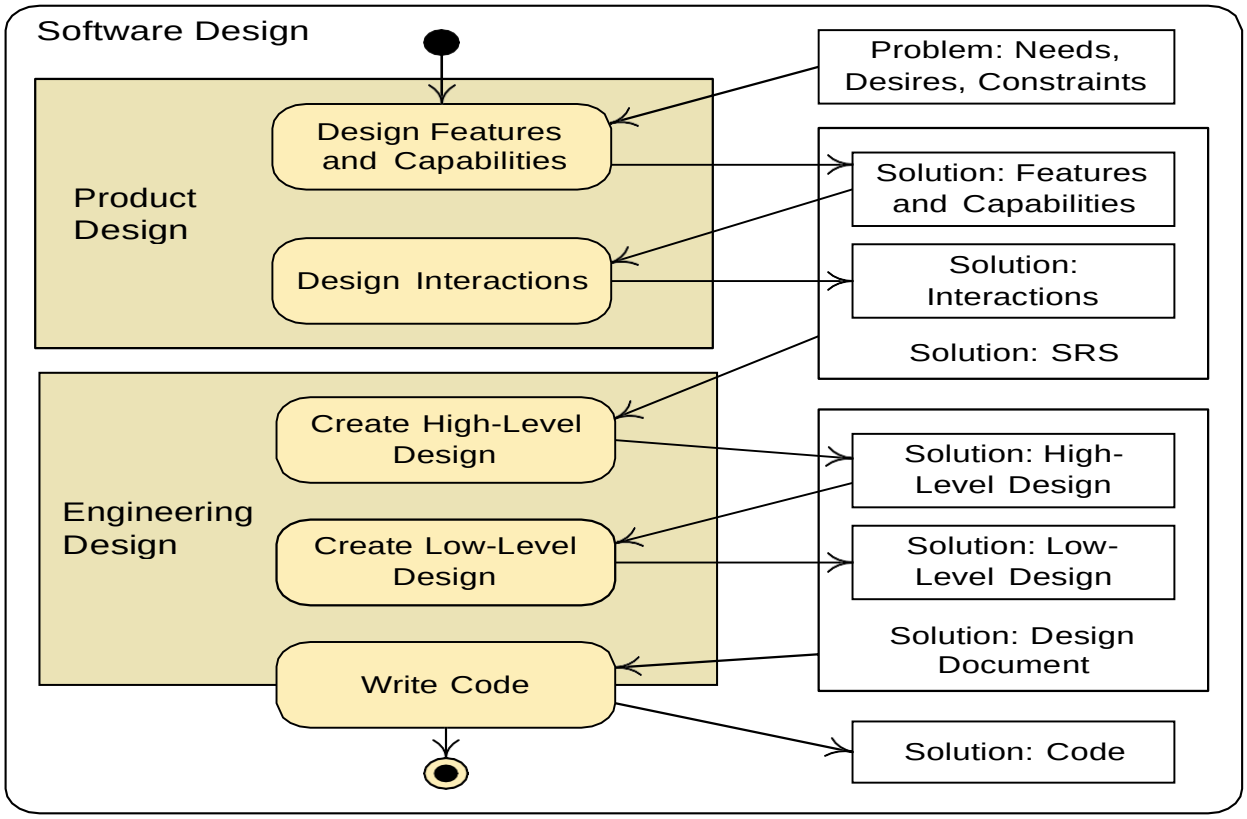
- Traditional way to make the distinction between requirements and design activities
- Not adequate because
  - Many "what" specifications turn out to be design decisions
  - Many "how" specifications turn out to be client or customer needs or desires

- Distinguish requirements from design based on problem solving: requirements activity formulates a problem solved in design

Design Problems and Solutions:

Problems and solutions demarcate various software design activities. Product design tackles a client problem and produces a product specification as a solution. This solution presents the problem to engineering designers, who produce a design document as their solution.

Design Problems and Solutions:



"Design" as a Verb and a Noun:

- This activity is what we refer to when we use the word "design" as a verb, as in the sentence "Engineers design programs meeting requirements specifications." But we have also used "design" as a noun, as in the sentence "Engineers develop a design meeting requirements specifications."

- Obviously, the word “design” is both a verb and a noun and refers to both an activity and a thing. A design specification is the output of the design activity and should meet the goals of the design activity—it should specify a program satisfying client needs and desires, subject to constraints.

Software Engineering Design Methods:

- A **software design method** is an orderly procedure for generating a precise and complete software design solution that meets client’s needs and constraints.

Design Method Components:

A method typically specifies the following items:

- ***Design Process*** —A collection of related tasks that transforms a set of inputs into a set of outputs
- ***Design Notations*** —A symbolic representational system
- ***Design Heuristics*** —Rules providing guidance, but no guarantee, for achieving some end
- Design methods also use **design principles** stating characteristics of design that make them better or worse.

History of Software Engineering Design Methods:

- The first design method was **stepwise refinement**, a top-down technique for decomposing procedures into simpler procedures until programming-level operations are reached**.**

- The dominant design methods from the mid-1970s through the early 1990s were various versions of **structured design.**
- Structured design methods focus on procedural composition but include other sorts of models as well.
- Object-oriented design methods emerged in the 1990s in response to shortcomings of structured design methods.
- Object-oriented methods promote thinking about programs as collections of collaborating objects rather than in terms of procedural decomposition.

Method Neutrality:

- Strongly emphasizes object-oriented notations, heuristics, and models.
- Most of the notations used in this course are UML notations, but some other important notations are included as well.
- A design task is a small job done in the design process, such as choosing classes or operations, or checking whether a model is complete. Notation and task heuristics are discussed throughout the course when notations and design tasks are introduced.

(WEEK 2 COMPLETED)

WEEK #3

Modeling processes with Activity diagram:

Modeling:

- A picture is worth 1000 words.
- A model is a representation of reality, like a model car, airplane.
- Most models have both diagrams and textual components.

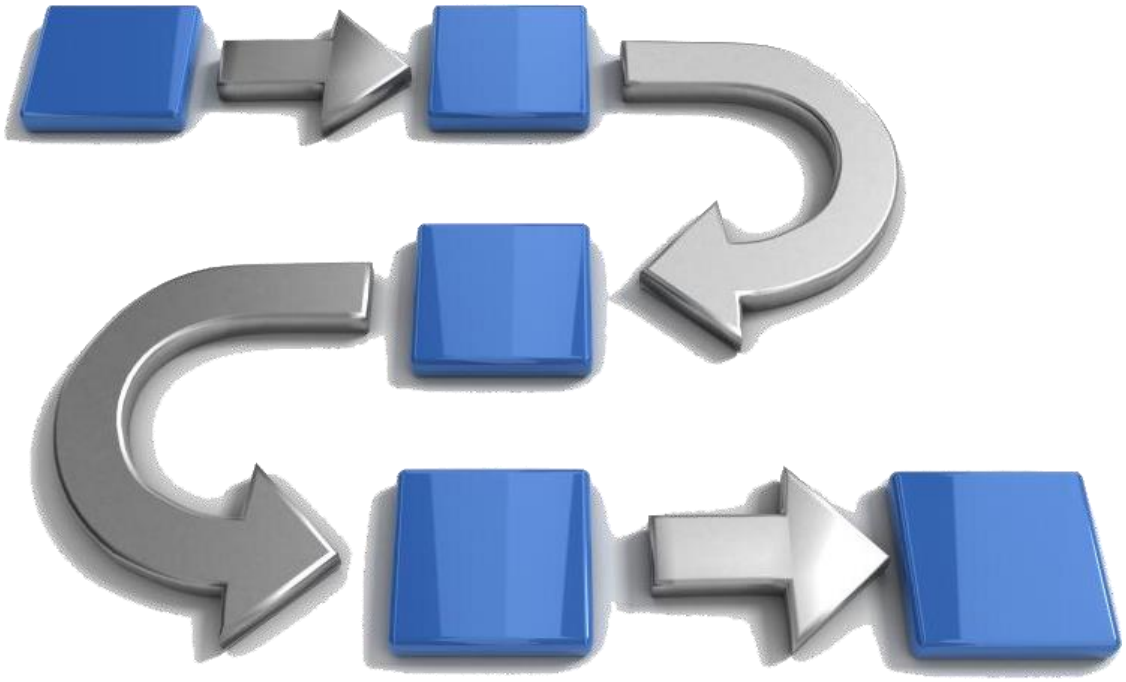
What is UML?

- UML stands for “Unified Modeling Language”

- It is an industry-standard graphical language for specifying, visualizing, constructing, and documenting the artifacts of software systems.
- The UML uses mostly graphical notations to express the OO analysis and design of software projects.
- Simplifies the complex process of software design.

Process:

A process is a collection of related tasks that transforms a set of inputs into a set of outputs.



Design Process:

- A design process is the core of any design endeavor, so it is essential that designers adopt an efficient and effective process.

- We need process description notations for design process.
- We will use UML Activity diagram.

Activity diagram:

An activity diagram shows actions and the flow of control and data between them.

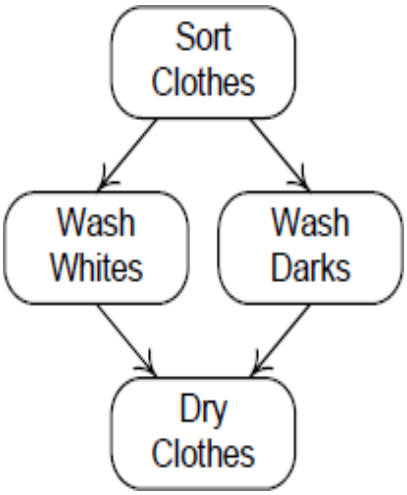
Activity, action and execution:

- An **activity** is a non-atomic task or procedure decomposable into actions.

- Shipping a product
- Wash clothes
- …

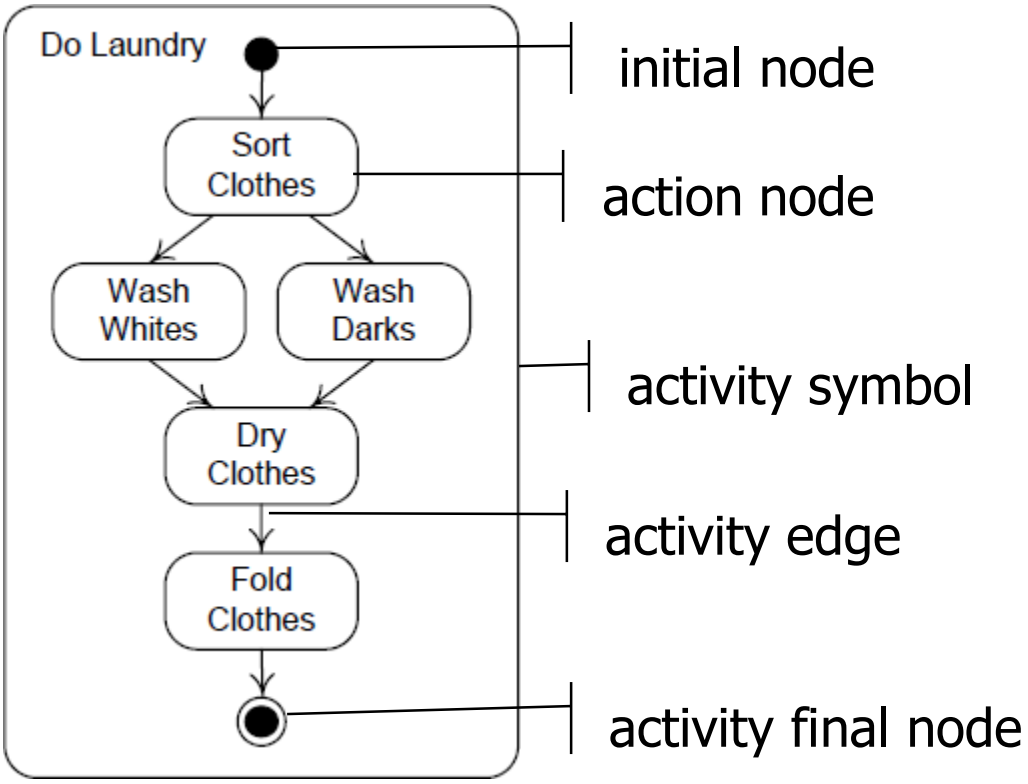
- An **action** is a task or procedure that cannot be broken into parts (i.e. it is atomic).
  - Check products in stock
  - Check dead level
  - Package the product

- **Activity diagrams model processes as an *activity graph*.**
  - *Activity nodes* represent actions or objects
    - Rounded rectangle containing arbitrary text naming or describing some action.
  - *Activity edges* represent control or data flows.
    - Represented by solid arrows with unfilled arrow heads.



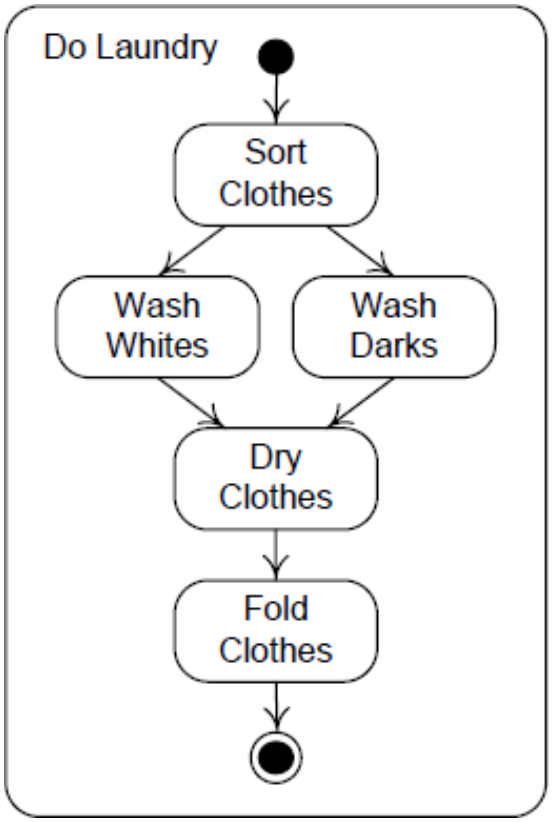
ACTIVITY DIAGRAM

Activity graph elements:

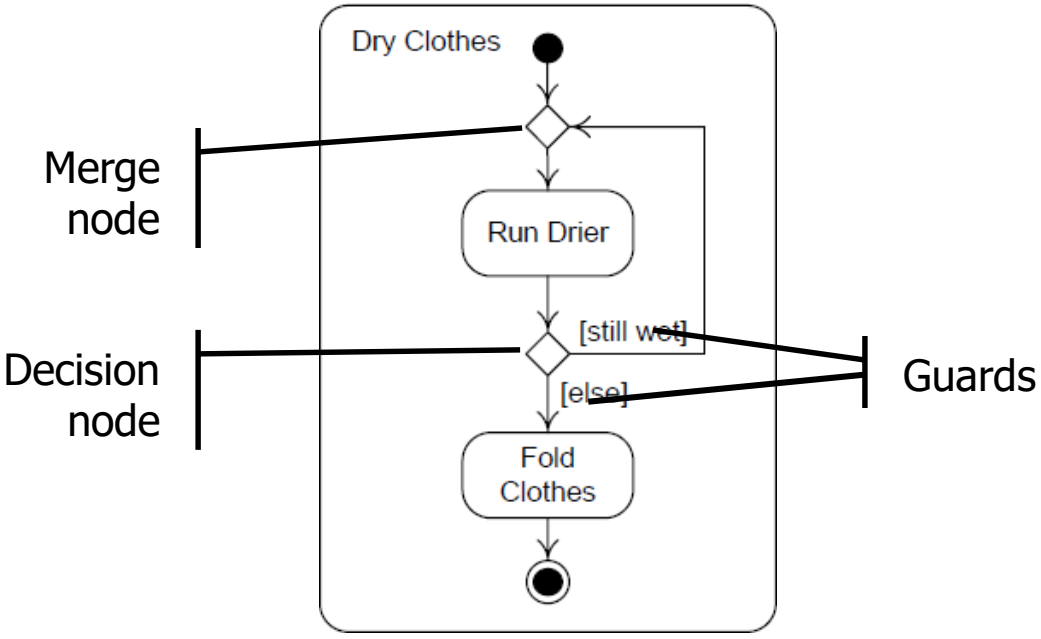


Activity diagram execution:

- Execution is modeled by tokens.
- When there is a token on every incoming edge of an action node, it consumes them and begins execution.
- When an action node completes execution, it produces tokens on each of its outgoing edges.
- An initial node produces a token on each outgoing edge when an activity begins.
- An activity final node consumes a token available on any incoming edge and terminates the activity.



Branching nodes:

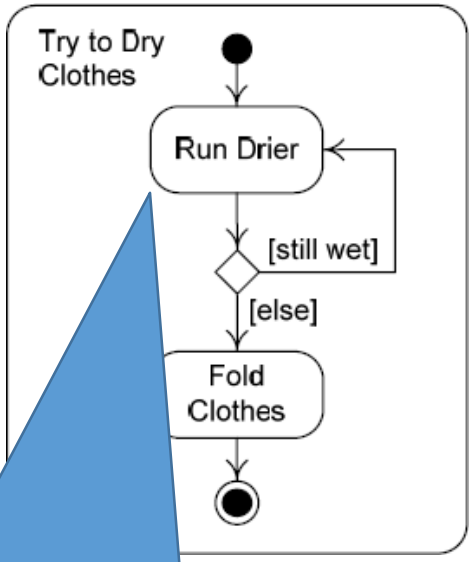


Branching execution:

- If a token is made available on the incoming edge of a decision node, the token is made available on the outgoing edge whose guard is true.
- If a token is available on any incoming edge of a merge node, it is made available on its outgoing edge.
- Guards must be mutually exclusive.

Deadlocks:

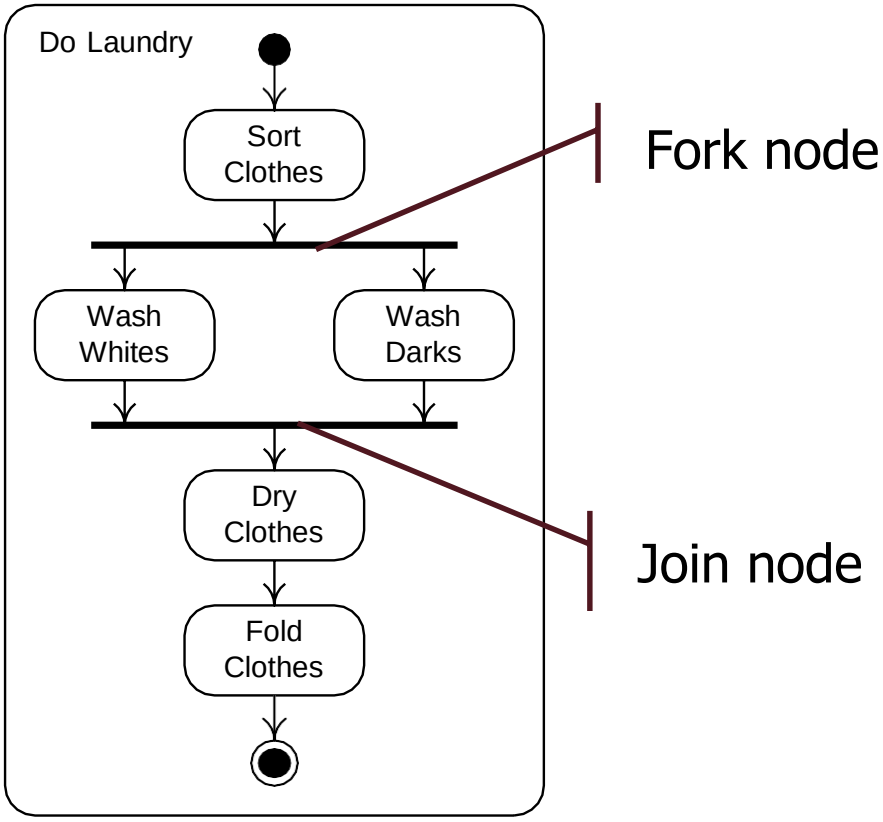
Run Drier cannot execute: when the activity begins, there is a token on the edge from the initial node but not on the other incoming edge.



When there is a token on every incoming edge of an action node, it consumes them and begins execution.

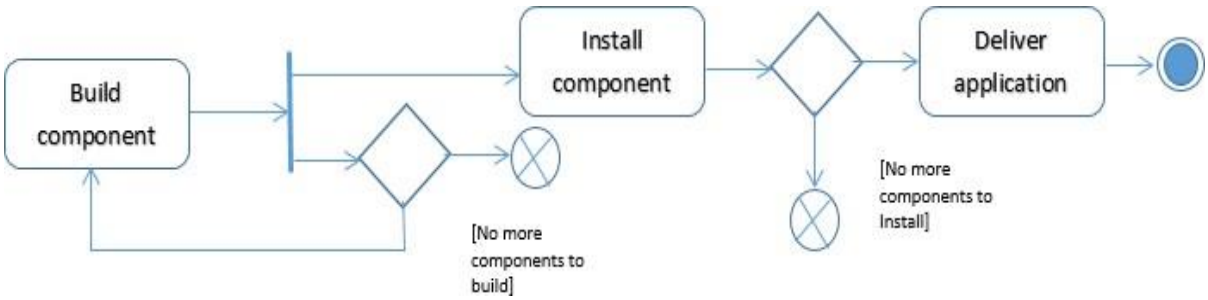
0xJG9xvDuSByHKTqyndvX

Forks and joins:



Forks and joins execution:

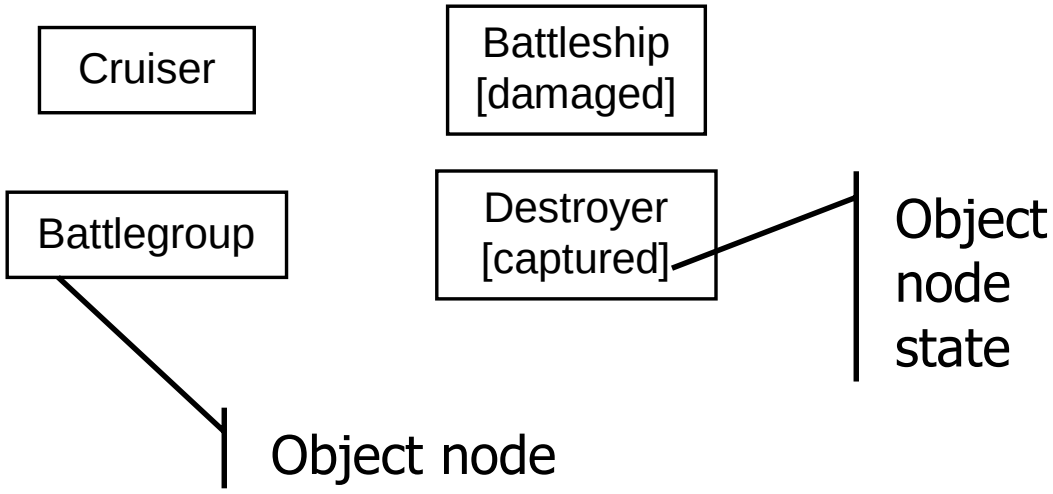
- A token available on the incoming edge of a fork node is reproduced and made available on all its outgoing edges.
- When tokens are available on every incoming edge of a join node, a token is made available on its outgoing edge.
- Concurrency can be modeled without these nodes.



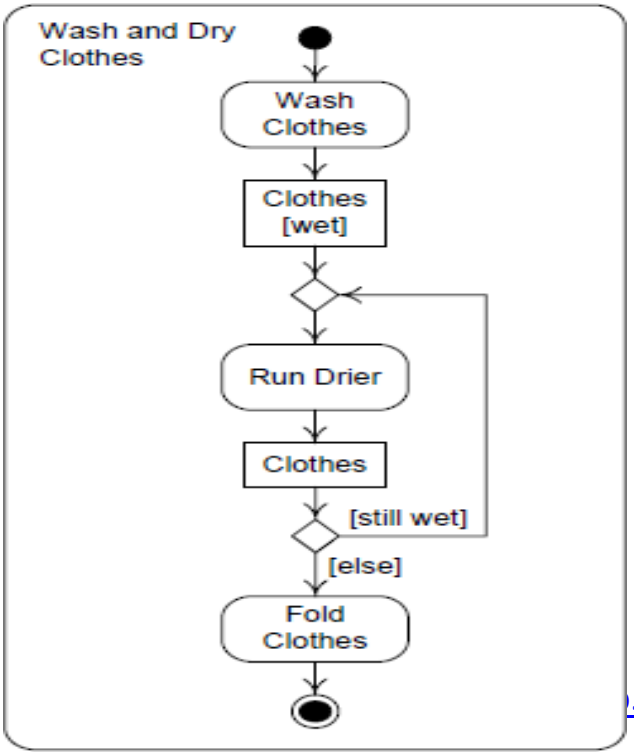
Object Nodes:

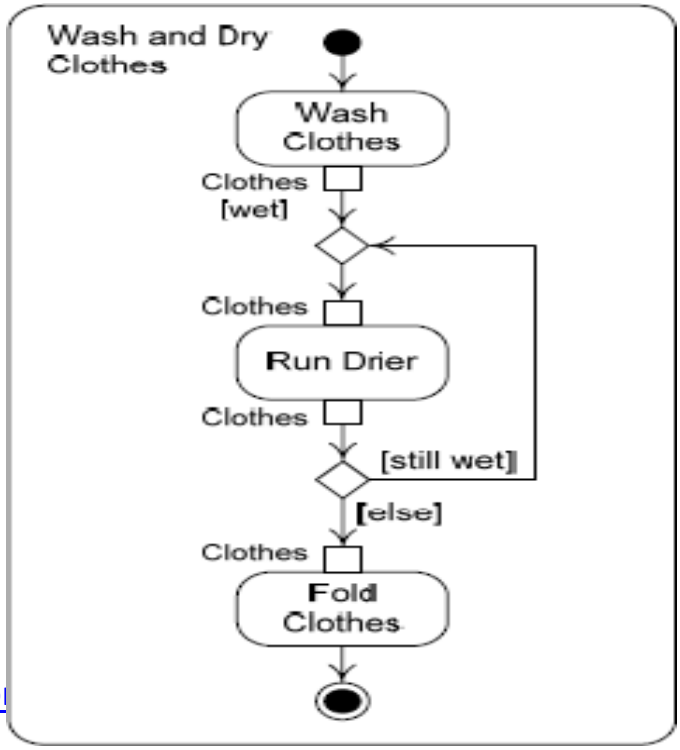
Data and objects are shown as object nodes.

Any flow that begins or ends at an object node is a data flow.



Object Nodes:





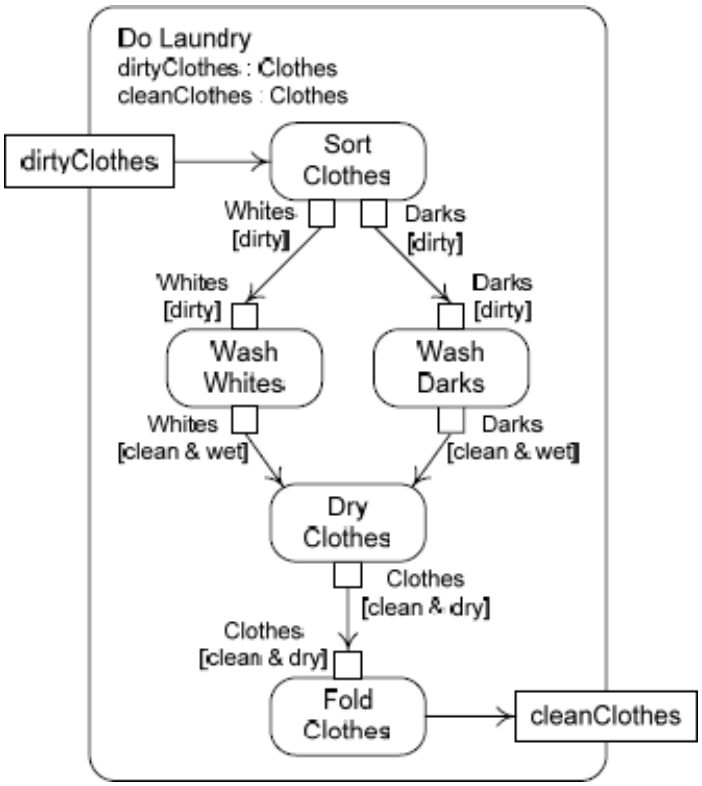
Control and Data Flows:

- Control tokens do not contain data, data tokens do.
- A control flow is an activity edge, conduit for control tokens.
- A data flow is an activity edge, conduit for data tokens.
- Rules for token-based execution apply just as well to data flows as to control flows, with the addition of a mechanism for adding and removing data from tokens.

Activity parameters:

- An activity parameter is an object node placed on the boundaries of an activity symbol to represent data or object inputs or outputs.
- Input activity parameters have only outgoing arrows, and output activity parameters have only incoming arrows.

Activity parameters (Example):



Activity diagram heuristics:

- Model flow control and objects down the page and from left to right.
- Name activities and action nodes with verb phrases.
- Name object nodes and pins with noun phrases.
- Don’t use both control and data flows when a data flow alone can do the job.
- Make sure that all flows entering an action node can provide tokens concurrently.
- Use the [else] guard at every branch.

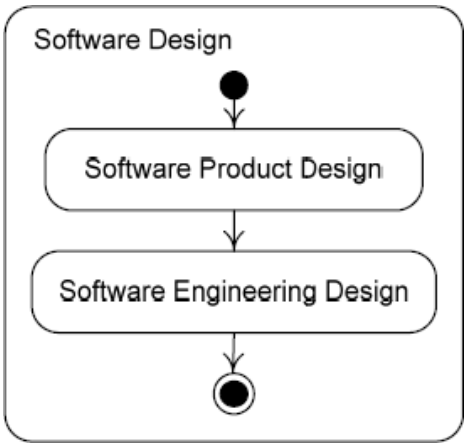
(WEEK 3 COMPLETED)

(WEAK # 4)

Software design processes

Software design:

- Software design consists of two different activities.
  - Software product design
  - Software engineering design

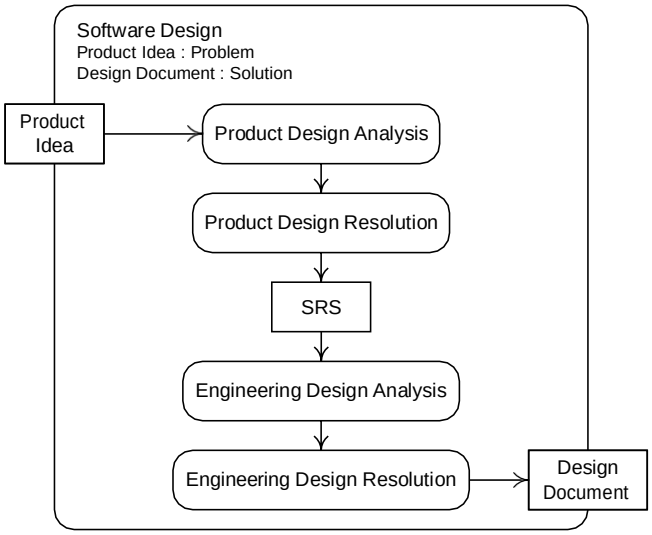


Analysis and resolution:

- The first step of "problem solving" must always be to understand the problem.
- If design is problem solving, then this activity must be the first step in design.
- *Analysis* is the activity of breaking down a design problem for the purpose of understanding it.
- Once problem is understood, next step is to solve it.
- Unfortunately the activity of solving a design problem does not have a good, widely accepted name.
- Traditionally this activity has been called design, but this is very confusing.
- In the traditional way of speaking, design consists of the following steps:
- *Analysis*—Understanding the problem.
- *Design* —solving the problem.
- In our context, we refer to the activity of solving a design problem as *resolution*.

- The terms used in our context will be:
  - **Analysis** is breaking down a design problem to understand it.
  - **Resolution** is solving a design problem

Analysis and Resolution in Software Design:



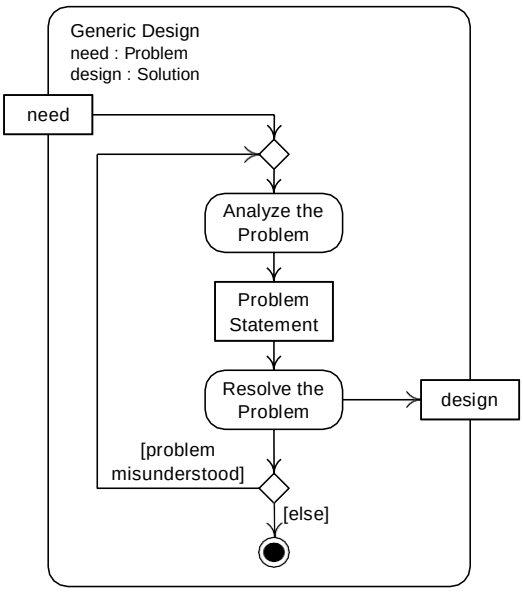
A generic problem solving strategy:

- Understand the problem
- Generate candidate solutions
- Evaluate solutions
- Select best solution(s)
- Iterate if no solution is adequate
- Ensure the solution is complete, well-documented, and deliver it

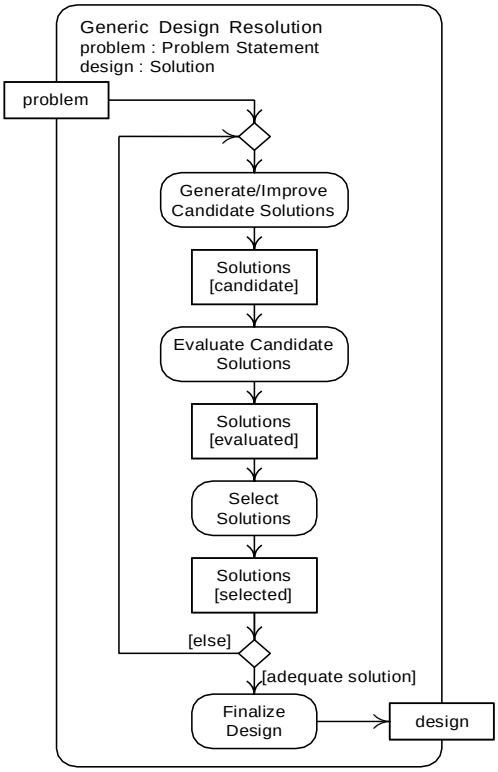
A generic design process:

- Analyze the Problem
- Generate/Improve Candidate Solutions
- Evaluate Candidate Solutions
- Select Solutions
- Iterate
- Finalize the Design

Generic design process:



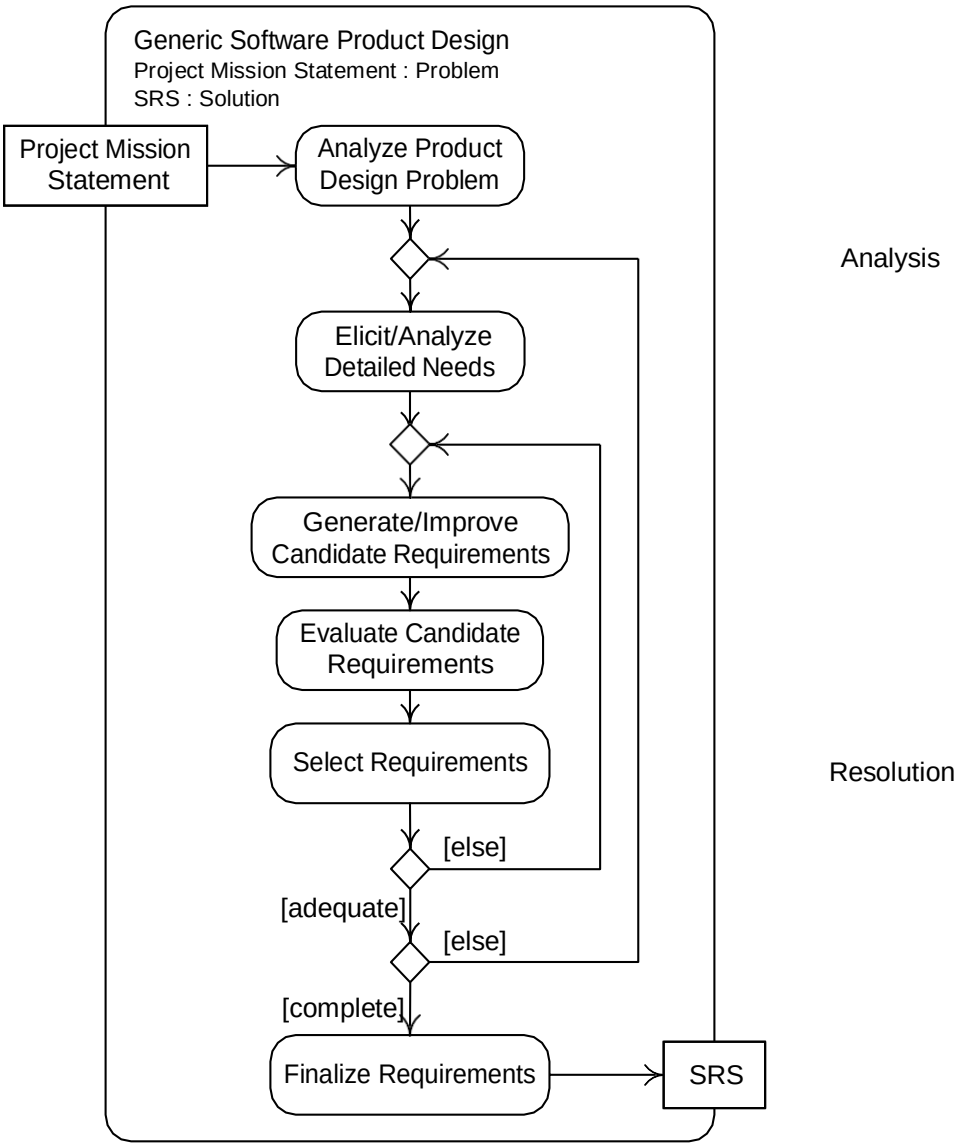
A Design Resolution Process:



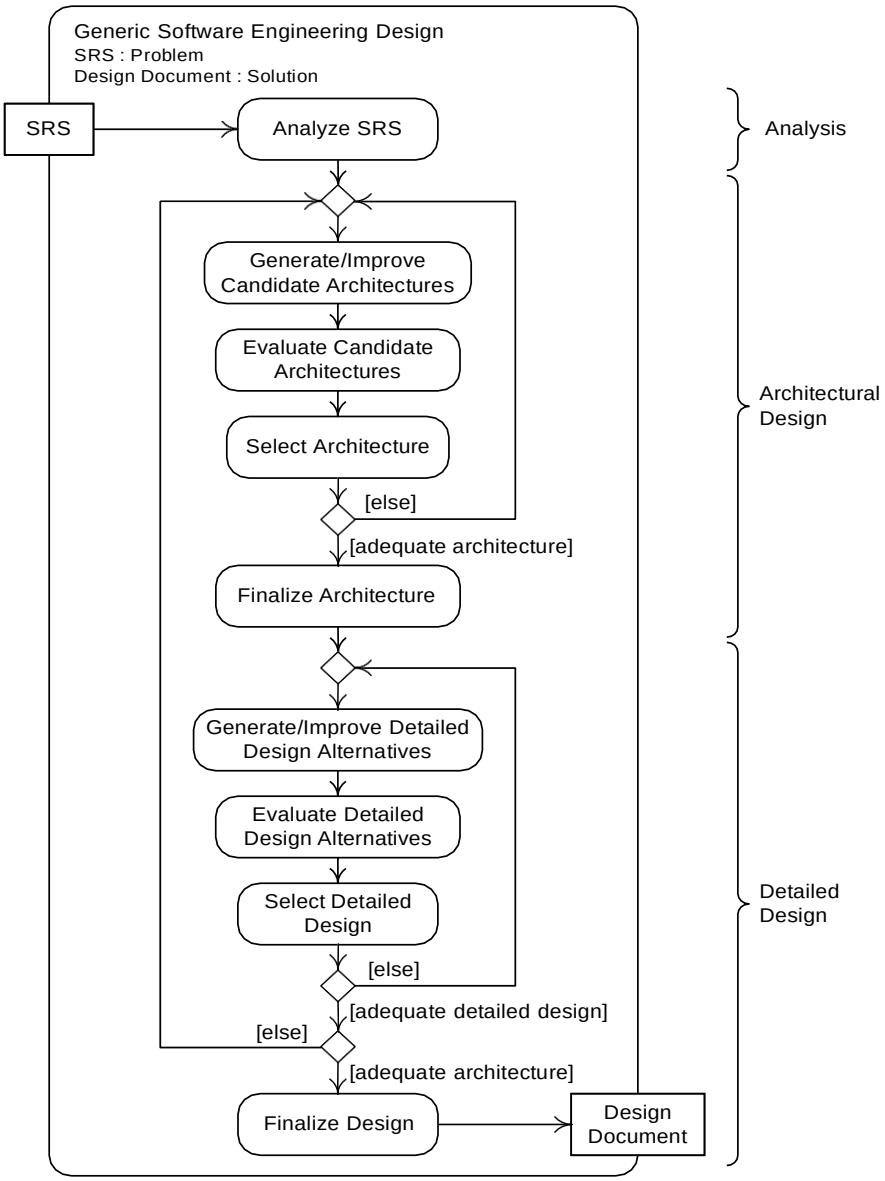
Design Process Characteristics:

- Designers should generate many candidate solutions during the design process.
- The design process is highly iterative; designers must frequently reanalyze the problem and must generate and improve solutions many times.

A Generic Software Product Design Process:



A Generic Software Engineering Design Process:



Software design management

Design require management:

- Software development is complex, expensive, time consuming done by groups of people.
- If it is simply allowed to "happen," the result is chaos.
- Chaos is avoided when software development is managed.
- Software development must be planned, organized, and controlled, and the people involved must be led.

There are at least two sorts of business activities that must be managed.

- **Operations** are standardized activities that occur continuously or at regular intervals.

    1) Hiring and performance review
    2) Payroll operations
    3) Shipping and receiving operations

- A **project** is a one-time effort to achieve a particular, current goal of an organization, usually subject to specific time or cost constraints.

    1) Efforts to introduce new products,
    2) Redesign tools and processes to save money
    3) Restructure an organization in response to business needs.

Project planning activities:

- Software development clearly fits project management:
- **Planning:** Formulating a scheme for doing a project.
- **Organizing:** Structuring the organizational entities involved in a project and assigning them responsibilities and authority.
- **Staffing:** Filling the positions in an organizational structure and keeping them filled.
- **Tracking**: Observing the progress of work and adjusting work and plans accordingly.
- **Leading**: Directing and helping people doing project work.

Project Planning:

- The first step in working out a project plan is to determine how much work must be done and the resources needed to do it.
- **Estimation** is calculation of the approximate cost, effort, time, or resources required to achieve some end.
    - Mostly begin by estimating the size of work products such as source code, documentation, and so forth, and then deriving estimates of effort, time, cost, and other resources.
- A **schedule** specifies the start and duration of work tasks, and often the dates of milestones.
- A **milestone** is any significant event in a project.
- A **risk** is any occurrence with negative consequences.
- **Risk analysis** is an orderly process of identifying, understanding, and assessing risks.
- The final portion of the project plan is a specification of various rules governing work. Such rules fall into the following categories:
    1) *Policies and Procedures*
    2) *Tools and Techniques*

Project organization:

- There are many ways to organize people into groups and assign them responsibilities and authority
    - Organizational structure.
- There are also many ways for people in groups to interact, make decisions, and work together
    - Team structures.

Organizational structure:

- **Project organization:** Groups might be responsible for carrying projects from their inception through completion
- **Functional organization**: Groups might be responsible for just part of the project, such as design or coding or testing

Team structure:

- ***Hierarchical team:*** A team might have a leader who makes decisions, assigns work and resolves conflicts.
- ***Democratic team:*** A team might attempt to make decisions, assign work, and resolve conflicts though discussion, consensus, and voting.

Project staffing:

- An organizational structure has groups with roles that must be filled e.g. testing group.
- Project staffing is the activity of filling the roles designated in an organizational structure and keeping them filled with appropriate individuals.
    - Hiring and orienting new employees
    - career development guidance
    - opportunities through training and education
    - Evaluating their performance

Project tracking:

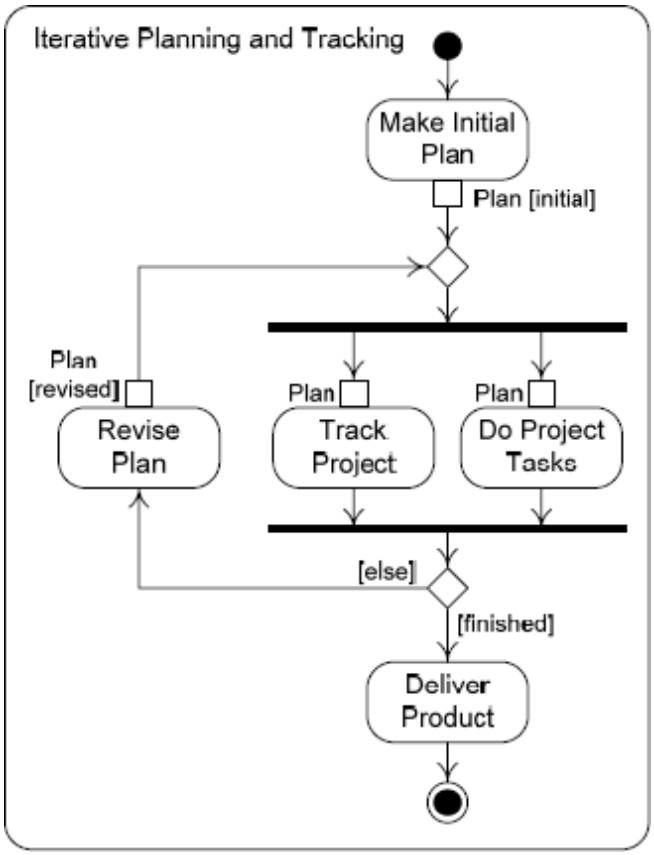
- Nothing ever goes exactly as planned, so it is essential to observe the progress of a project and adjust the work, respond to risks, and, if necessary, alter the plan.
- **Project Tracking**: Measuring and reporting the status of milestones, tasks and activities required in achieving the pre-defined project results

- Reasons for project tracking:
    - A task may simply take more or less time than expected.
    - Some of the rules governing the project may cause problems.
    - The resources needed to accomplish tasks may not be as anticipated.
    - Something bad may occur.
    - Tracking is essential so that estimates, schedules, resource allocations, risk analyses, and rules can be revised.

Leading a project:

- An adequate direction and support, a broad category of management responsibility called **leadership** is required for successful project.
- Merely directing people does not guarantee success.
    - People also need a congenial work environment, an emotionally
    - socially supportive workplace,
    - Make them feel that they are doing something important

Iterative Planning and Tracking:



(WEEK 4 COMPLETED)

WEEK # 5

Software design management

Project planning activities:

- Software development clearly fits project management:
- **Planning:** Formulating a scheme for doing a project.
- **Organizing**: Structuring the organizational entities involved in a project and assigning them responsibilities and authority.
- **Staffing**: Filling the positions in an organizational structure and keeping them filled.
- **Tracking**: Observing the progress of work and adjusting work and plans accordingly.
- **Leading**: Directing and helping people doing project work.

Design Project Decomposition:

- Most aspects of project management depend on the work to be done and, in particular, on how it is decomposed.
- An obvious way to break down a design project is to divide the work according to the generic design processes discussed in the last section.

Work Phase		Typical Work Products
Product Design	Analysis: Design Problem	Statement of interested parties, product concept, project scope, markets, business goals Models (of the problem) Prototypes (exploring the problem)
	Analysis: Detailed Needs	Client surveys, questionnaires, interview transcripts, etc. Problem domain description Lists of needs, stakeholders Models (of the problem) Prototypes (exploring needs)
	Resolution: Product Specification	Requirements specifications Models (of the product) Prototypes (demonstrating the product)

Work Phase		Typical Work Products
Engineering Design	Analysis	Models (of the engineering problem) Prototypes (exploring the problem)
	Resolution: Architectural Design	Architectural design models Architectural design specifications Architectural prototypes
	Resolution: Detailed Design	Detailed design models Detailed design specifications Detailed design prototypes

Design Project Planning:

- The initial project plan focuses on design problem analysis, with only rough plans for the remainder of the work.
- So, plan will be revised before product design resolution, engineering design analysis, and engineering design resolution.
- Initial estimates of effort, time, and resources are as precise as possible, based on the work products to be completed.
- These estimates may be based on data about work done in the past or an analogy with similar jobs with which the planners are familiar.
- The estimates are then used to block out:
  - An initial schedule
  - Allocate resources
  - Analyze risks
  - Set the rules guiding the project.

Design Project Tracking:

- Product analysis work is tracked against the initial plan.
- Ideally, problem analysis is complete when it is time to revise the plan, since planning the product design resolution phase requires this information.
- The plan may be altered during tracking to make this happen.
- A revised plan prepared before the product design resolution phase should have much more accurate:
  - Estimates
  - Schedule
  - Resource allocations
  - Risk analysis

Iterative planning and tracking continues through the engineering design with more details added each time the plan is revised.

Design Project Organization:

- Design teams should be formed with responsibility of:
  - Design as a whole
  - Each major phase
  - Each sub-phase
  - Production of the various work products.
  - e.g. a large company might have a division responsible for requirements and design.

Design Project Staffing:

- Organizations are staffed to fit the decomposition of design work.
- Projects need staff to:
  - Elicit and analyze needs
  - Create prototypes
  - Model systems
  - Create product designs
  - Write requirements specifications
  - Design user interaction
  - Make high-level and low-level engineering designs
  - Quality assurance.

Design Project Leadership:

- Leading a design problem needs extra skills:
  - Visionary
  - Creative
  - Anticipate changes
  - Experience

Design as project driver:

- Design work extends from the start of a software development project to the coding phase, and it recurs during maintenance.
- Two major products of software design, the SRS and the design document, are the blueprints for coding and testing.
- So, design is the driving activity in software development.
- By the time the software design is complete, enough information is available to make accurate and complete plans for the coding and testing phases.
- Good design work early in the life cycle is crucial for software development project success.

Context of Software Product Design

Products and markets:

- Organizations create products for economic gain.
- Product development is very expensive, so an organization must be careful to create products that it can actually sell or use.

A market is a set of actual or prospective customers who need or want a product, have the resources to exchange something of value for it, and are willing to do so.

Importance of market:

Organizations study markets to:

- Choose which markets to sell to (**target markets**)
- Choose what products to develop
- Determine product features and characteristics

Thus, the sorts of products that an organization decides to develop ultimately depend on the target markets to which it hopes to sell the products.

Products influence design:

- A lot of what happens during product design depends on what sort of product is being designed.
- A product's characteristics influences:
  - The decision to develop the product;
  - The resources and time devoted to product development;
  - The techniques
  - Methods, and tools used to develop the product;
  - Distribution and support of the final product.

Categorizing products:

- Products fall into different categories along several dimensions.
- A **product category** is a dimension along which products may differ.
  - Target market size
  - Product line novelty
  - Technological novelty
- A **product type** is a collection of products that have the same value in a particular product category.

Target Market Size:

Target market size is the number of customers a product is intended to serve.

Type	Description	Examples
Consumer	Mass consumer markets	Word processors, spreadsheets, accounting packages, computer games, operating systems
Niche Market	More than one customer but not a mass consumer market	Programs for configuration management, shipyard management, medical office records management, AquaLush
Custom	Individual customers	Systems written for one part of a company by another part, space shuttle software, weapons software

Categorizing products:

- Designers of custom and niche-market products designers can identify needs and desires for a product as compared to consumer products.
- Designing consumer products is easy than designing niche-market products which is easy than designing custom products.
- Competitors are important when designing consumer and niche-market products, but this is not the case designing custom products.
- Different aspects of product design are more or less important in these different categories.

1) Consumer products place a premium on attractive user interface design.
2) Functionality is usually more important for custom and niche-market products.

Product Line Novelty:

- **Product line novelty** is how "new" a product is in relation to other products in current product line.

Type	Description	Examples
New	Different from anything else in the product line	Tax preparation product in a line of accounting products, AquaLush
Derivative	Similar to one or more existing products in the product line	Database management system for individual users in a line of systems for corporate users
Maintenance Release	New release of an existing product	Third release of a spreadsheet

Product Line Novelty:

- Maintenance releases pose higher constraints on designers than derivative products which pose higher constraints than designing new products.
- Designing a new product is a very big job, designing a derivative product is a smaller but still formidable task, and designing a new release may be relatively easy.

Technical Novelty:

Technical novelty means "how much new technology" is incorporated in a product, w.r.t. target market at a particular time.

Type	Description	Examples
Visionary Technology	New technology must be developed for the product	Mobile computing (2000), Wearable automatic lecture note-taker (2004)
Leading-Edge Technology	Proven technology not yet in widespread use	Peer-to-peer file-sharing products (2002), AquaLush (2006)
Established Technology	Widely used, standard technology	Products with graphical user interfaces (2000)

Technical Novelty:

- Designing products with visionary or leading-edge technologies is most difficult kind of software design.
    - Hard to figure out what clients want.
    - Whether products with new technology will attract customers
- Products with visionary technology may never be built if efforts to develop the new technology fail.
- Even if they are a technological success, they may still fail in the marketplace if customers don't like the new technology.
- Leading-edge and established technology products are more likely to succeed.

(WEEK 5 COMPLETED)

(WEEK # 6)

Project Mission Statement:

- A project mission statement is a document that defines a development project's goals and limits.
- The project mission statement plays two important roles:
  1) Launches a development project
  2) States the software design problem
- The project mission statement is the main input to the product design process.

Project Mission Statement Template:

1. Introduction
2. Product Vision and Project Scope
3. Target Markets
4. Stakeholders
5. Assumptions and Constraints
6. Business Requirements

Introduction:

- The introduction contains background information to provide context.
- Information about the major business opportunity that the new product will take advantage of and the product operating environment.

Product Vision and Project Scope:

- A product vision statement is a general description of the product's purpose and form.
- The project scope is the work to be done on a project.
  - Often only part of the product vision.
  - May list what will *not* to be done as well as what will be done.

Target Market:

- Upper management chooses the target market segments for a new product or release during product planning.
- Target markets are those market segments to which the organization intends to sell the new product. Market segments determine users, features, competitors, and so forth.

Stakeholders:

- A stakeholder is anyone affected by a product or involved in or influencing its development.
  - Product users and purchasers
  - Developers and their managers
  - Marketing, sales, distribution, and product support personnel
  - Regulators, inspectors, and lawyers
- Developers must know the target market and stakeholders to build a product satisfying stakeholders' needs.

Assumptions and Constraints:

- An assumption is something that developers take for granted.
  - Feature of the problem
  - Examples: target deployment environments, levels of user support
- A constraint is any factor that limits developers.
  - Restriction on the solution
  - Examples: cost and time limits, conformance to regulations

Business Requirements:

A business requirement is a statement of a client or development organization goal that a product must meet.

- Time, cost, quality, or business results
- Should be stated so that it is clear whether it is satisfied (quantitative goals)
- Broad goals related to business, not detailed product specifications

Needs Elicitation

Needs VS Requirements:

- Stakeholder needs and desires define the product design *problem*.
- Requirements specify the product design *solution*.
- Needs and requirements statements are similar, but the heart of product design is moving from needs to requirements.
    - Conflicting needs and desires
    - Tradeoffs (needs and constraints)
    - Ways of satisfying needs and desires

Needs Elicitation Challenges:

- Stakeholders often cannot explain their work, or articulate their needs and desires.
- Needs and desires can only be understood in a larger context that includes understanding the problem domain.
- Stakeholders make mistakes, leave things out, and are misleading.
- Stakeholders often don't understand the capabilities and limitations of technology.
- Designers are faced with a flood of information, often contradictory, incomplete, and confusing.

How to tackle Elicitation Challenges:

- Designers must obtain information from stakeholders in a systematic fashion using several elicitation techniques and must document and analyze the results to ensure that needs and desires are understood correctly and completely.
- The main way to organize requirements elicitation is to work from the top down through levels of abstraction. Organization within each level of abstraction is achieved by focusing on particular product aspects, which depend on the product itself.

Elicitation Heuristics:

- **Learn about the problem domain first.**

If designers don't understand the problem domain, they need to elicit, document, and analyze information about it *before* eliciting needs.

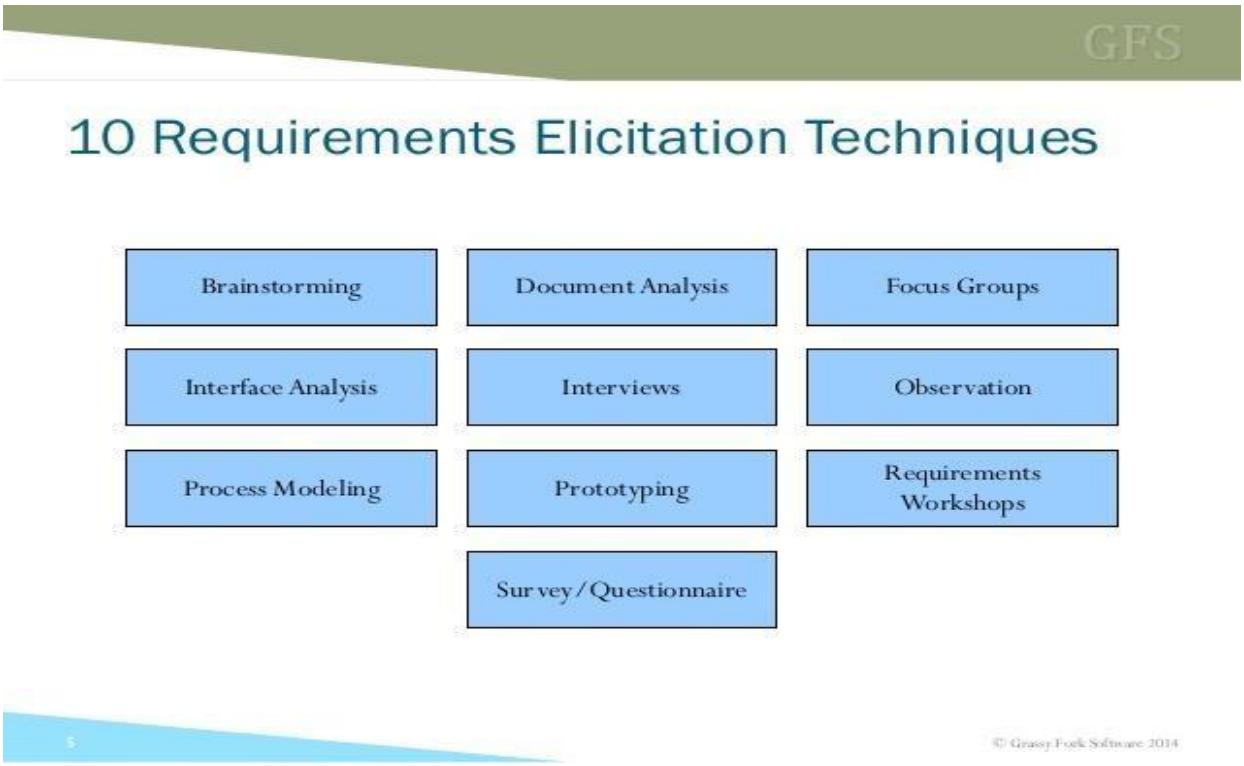
- **Determine stakeholder goals as the context of stakeholder needs and desires**

What a stakeholder needs and wants is a consequence of his or her goals. For example, a user may need a product to record sample data. Why would the user need this? Because the user's goal is to monitor a manufacturing process by sampling and analyzing its output

- **Study user tasks**.

For example, suppose users currently collect and measure samples by hand, record the data in a log book, use a calculator to compute statistics, enter the results on a paper graph, and study the graph to see if the process is running properly.

Elicitation Techniques:



Elicitation Techniques:

- **Interviews:** Question and answer session during which one or more designers ask questions of one or more stakeholders or problem-domain experts
  - Most important technique for recording responses
- **Observation:** Many products automate or support work done by people, so designers need to understand how people do their work to design such products
  - Especially useful for eliciting derivative product and maintenance release needs because it can reveal many opportunities for product improvement
- **Focus Groups:** Is an informal discussion among six to nine people led by a facilitator who keeps the group on topic. Focus groups consist of stakeholders or stakeholder representatives who discuss some aspect of the product.
  - Main technique of obtaining needs for consumer products, especially new products and those with visionary or leading-edge technologies.
- **Prototype:** A working model of part or all of a final product. Prototypes provide a useful basis for conversations with stakeholders about features, capabilities, and user interface issues such as interaction protocols.
  - Especially useful for products with visionary technology because they help people understand what a product with the new technology will be like.
- **Questionnaires –** It is efficient technique to elicit information from many people.
  - Close ended questions

    Easier to analyze and range of possible responses is well-understood
  - Open ended questions

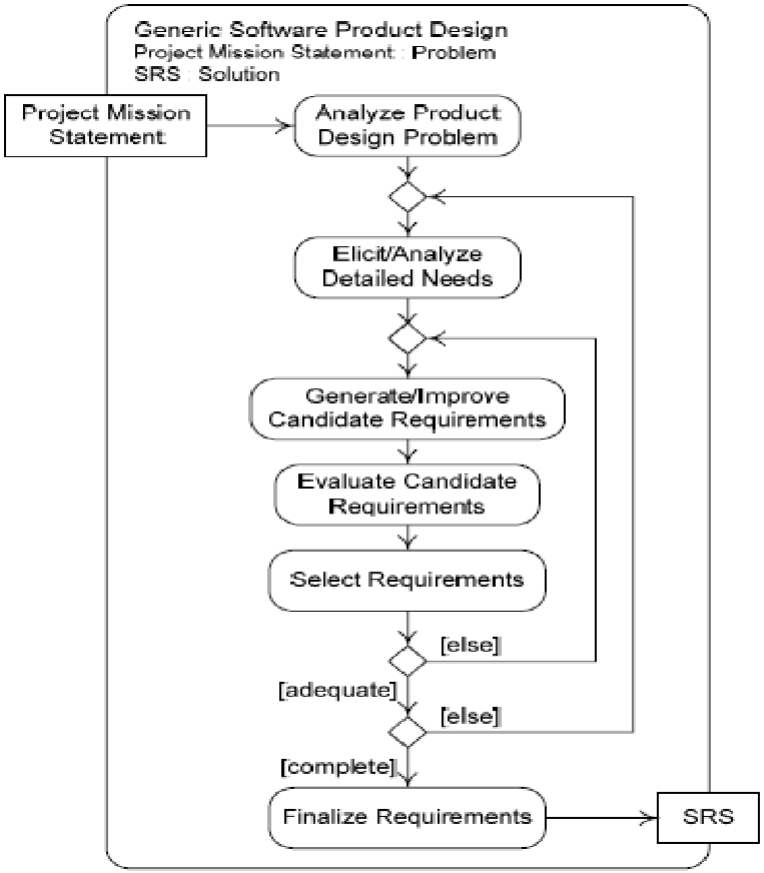
    It includes detailed responses and relatively harder to understand

---

(WEEK 6 COMPLETED)

WEEK # 7

Product Design Process Overview



Steps of Software Design Process:

There are six steps of software design process:

- Understanding of Design problem
- Elicit/Analyze Detailed Needs
- Generating/Improve Candidate Requirements
- Evaluate Candidate Requirements
- Select Requirements
- Finalize Requirements

1. **Understanding of Design Problem:**

   - The nature of this task depends on whether there is an adequate project mission statement
   - A good project mission statement defines the product design problem, so the designers need only study the mission statement and research any parts of it they do not understand.

2. **Elicitation of Detailed Needs**

   - Second step in design process is comprised of eliciting and analyzing detailed needs
   - Designers needs to learn much more about stakeholder needs and desires, especially those of users and purchasers that will meet its business requirements.

3. **4-5 Improvement, Evaluation and Selection of Requirements:**

   - Third step proceeds by generating and refining requirements, therefore fulfilling the needs determined during analysis
   - Once alternative requirements are generated and stated , they are evaluated in fourth step
   - In fifth step, requirements are selected on basis of evaluation

6. **Requirements Finalization:**

   - The last step of the software product design process is to finalize the SRS

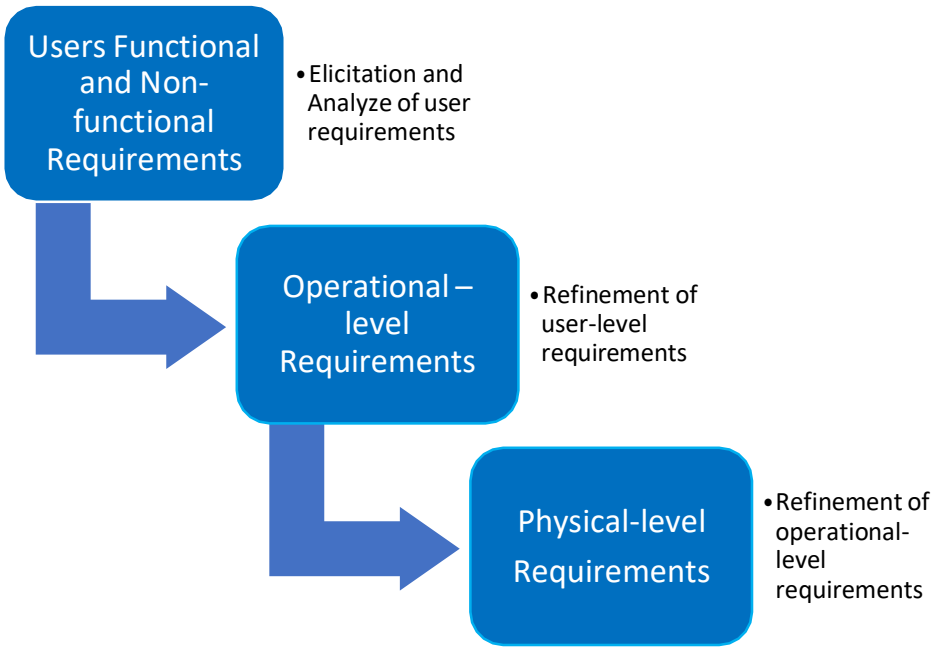
So we start with Project Mission statement that act as input to Design process and the outcome of this process is SRS (Software Requirement Specification).

Outer iteration in Fig 1 reflects refinement activity of product details specification.

Product Design Process: A Top-Down Process:

- Product design resolution sets technical requirements at a high level of abstraction and then refine them until all product details are specified.
- During this process, user-level needs are elicited and analyzed first, and user-level functional, data , and non-functional requirements are generated , refined, and evaluated until they are adequate
- The user-level requirements provide an abstract solution to the design problem. They are then refined to produce operational–level requirements
- Operational-level requirements are refined to produce physical-level requirements

Refinement Process:



Refinement is complete when these 3 level requirements are specified

Product Design Process: A User Centered Approach:

User-centered design comprises the following three principles:

- *Stakeholder Focus* – Determine the needs and desires of all stakeholders(especially users), and involve them in evaluating the design and perhaps even in generating the design
- *Empirical Evaluation* – Gather stakeholder needs and desires and assess design quality by collecting data rather than by relying on guesses.
- *Iteration* – Improve designs repeatedly until they are adequate.

Terminologies:

- **Requirement Elicitation –** Collecting stakeholder needs and desires is called *requirements elicitation* , or *needs identification* or *needs elicitation*
- **Requirements Analysis –** Understanding stakeholder needs is called *needs analysis* or *requirements analysis*
- **Requirements Validation –** Confirming with stakeholder that a product design satisfies their needs and desires is called *requirements validation* or just *validation.*

Role of Stakeholders:

Activity	Stakeholders' Role
Analyze Product Design Problem	Clarify project mission statement Answer questions
Elicit Needs	Answer questions Be subjects of empirical studies
Analyze Needs	Answer questions Review and validate models and documents Participate in analysis with designers
Generate/Improve Alternatives	Participate in generation and improvement
Evaluate Alternatives	Answer questions Be subject of empirical studies Participate in evaluation with designers
Select Alternatives	Participate in selection with designers
Finalize Design	Review and validate requirements

Table 1: Stakeholders' Roles in product Design

Needs Documentation and Analysis

Formulating & Organizing Documentation:

- The raw data collected from interviews, observation, focus groups, workshops, competitive studies and so forth needs to be sorted, stated clearly, and organized.
- First step is to divide the data into two categories:
  1) Data about the problem domain
  2) Data about Stakeholders' goals, needs, and desires

Documenting the Problem Domain:

- Data about the problem domain can be further categorized and grouped to form an organized set of notes.

  1) Problem Domain Glossary is a useful tool in understanding the domain. Most problem domains have their own terminology that designers must learn

  2) Organization Chart can be made to display data about the stakeholders' organization.

Continue UML Activity Diagrams:

UML Activity Diagrams are useful tools for organizing and documenting problem domain information about business processes or user processes.

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. Data about processes obtained from interviews, observation, focus groups, or document studies can be represented in an activity diagram much better than in text.

Documenting Goals, Needs and Desires:

- Raw data about stakeholders' goals , needs and desires can be organized into two lists:

  1) A stakeholders-goal list

  2) Needs list

- A stakeholders-goals list is a catalog of important stakeholder categories and their goals.

B.1 AquaLush Irrigation System Overview

Introduction

MacDougal Electronic Sensor Corporation (MESC), an electronic sensor manufacturer, has decided to start a new company to exploit a newly perfected soil moisture sensor. The company, called Verdant Irrigation Systems (VIS), will develop and market lawn and garden irrigation systems.

Timers regulate most irrigation or sprinkler systems: They release water for a fixed period on a regular basis. This may waste water if the soil is already wet or not provide enough water if the soil is very dry. VIS products will use the new soil moisture sensors to control irrigation. Irrigation will still take place on a regular basis, but now it may be skipped if the soil is already wet or continued until the soil is sufficiently moist.

VIS's first product is the AquaLush Irrigation System. It is targeted at high-end residential or small commercial properties.

A small team is charged with developing the software driving this product.

Opportunity Statement

Create an irrigation system that uses soil moisture sensors to control the amount of water used.

Stakeholder Category	Goals
Purchasers	Pay the least for a product that meets irrigation needs
	Purchase a product that is cheap to operate
	Purchase a product that is cheap to maintain
Installers	Have a product that is easy and fast to install
Operators	Irrigation can be scheduled to occur at certain times
	Irrigation schedules can be set up and changed quickly
	Irrigation schedules can be set up and changed without having to consult instructions
Maintainers	It is quick and easy to tell when the product is not working properly
	It is quick and easy to track down problems
	It is quick and easy to fix problems
	The product is able to recover from routine failures (such as loss of power or water pressure) by itself
	One sort of failure (such as loss of power or water pressure) does not lead to other failures (such as broken valves or sensors)
	The product and its parts have low failure rates

A need statement documents a single product feature, function, or property needed or desired by one or more stakeholders.

- A need statement should
  1) Name the stakeholder category or categories
  2) State one specific need
  3) Be a positive declarative sentence

- Often requires interpretation of raw data

Here is a list of "Elicited Needs" and "Needs Statements":

Elicited Responses	Need Statements
The usual way to call up product information is for the customer to read the number out of the catalog.	Sales personnel need to retrieve product information using catalog identifiers.
It would be great if I could just click on the product number in the invoice and have the product information pop up in another window.	Sales personnel need to retrieve product information from customer order displays.
We do monthly and quarterly reports where we analyze how often customers request information about products.	Marketing analysts need reports about the frequency with which information is requested about each product.
I need to know how often my people are accessing product information.	Sales managers need reports about the frequency of system use by each salesperson.
Somebody has got to keep this data up-to-date. You know, we change about 20% of our stuff in every catalog.	Technical support personnel need to create, delete, update, retrieve, and display product descriptions in the product description database.
I often have trouble finding things about my account on the current Web site. I've seen sites that keep a big list of links in the left part of the screen, and that works pretty well.	Users need better Web site organization and navigation aids.

Table: Elicited Needs and Needs Statements

Problem Modeling:

- Many kinds of models can represent the problem and help designers understand it.
- Models document the problem, can be reviewed with stakeholders for consistency.
- Many modeling notations and techniques are useful for analysis like
  - Various UML diagrams
  - Use case descriptions, user interface diagrams, dialog maps

Checking Needs Documentation:

- **Correctness:** A statement is **correct** if it is contingent and accords with the facts.
- **Scope**: A goal or need is within the project scope if it can be satisfied using the planned features of the product created by the project.
- **Terminological Consistency:** Terminological consistency is using words with the same meaning and not using synonyms.
- **Uniformity**—A description has uniformity when it treats similar items in similar ways.
- **Completeness**—Documentation is complete when it contains all relevant material.

Review Activities

- Developers should use checklists
- Stakeholders should review documents

A Needs Documentation Checklist:

Correctness ☐ Every stakeholder category in the stakeholders-goals list represents a group of legitimate stakeholders. ☐ Every need statement in the needs list accurately reflects the purported stakeholder's need. ☐ All need statement priorities are correct.
Scope ☐ All stakeholder goals are within the project scope. ☐ Every stated need is within the project scope.
Terminology ☐ Every specialized term is defined in the problem domain glossary. ☐ Specialized terms are used as defined in the problem domain glossary. ☐ Terms are used with the same meaning throughout. ☐ No synonyms are used.
Uniformity ☐ All need statements are at similar levels of abstraction. ☐ Similar items are treated in similar ways.
Completeness ☐ Every important stakeholder category is included in the stakeholders-goals list. ☐ Every relevant stakeholder goal is recorded in the stakeholders-goals list. ☐ Every stakeholder goal is satisfiable by needs in the needs list. ☐ Every need in the needs list is necessary to reach some stakeholder's goals. ☐ Every entity mentioned in the glossary and the needs list is included in the models. ☐ All needed operations are listed for every entity.

(WEEK 7 COMPLETED)

(WEEK # 8)

Software Requirement Specification (SRS)

Software Requirement:

- A software (product) requirement is a statement that a software product must have a certain feature, function, capability, or property.
- Requirements are captured in specifications, which are simply statements that must be true of a product.

Software Requirement Specification:

A **software requirements specification (SRS)** is a document cataloging all the requirements for a software product.

- This activity is aimed at finding out from the product's intended clients, and other interested parties, what they need and want from a software product. These needs and desires are translated into specifications of the functions, capabilities, appearance, behavior, and other characteristics of the software product.

- These specifications constitute the software product requirements, and they are recorded in a **software requirements specification (SRS)** document.

SRS in Software Product Life Cycle:

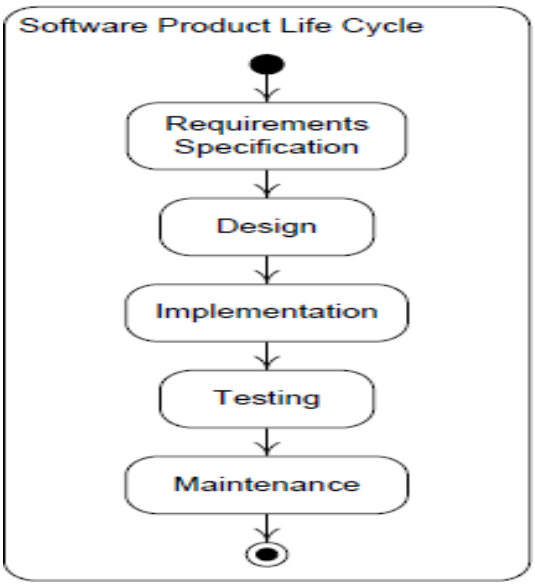
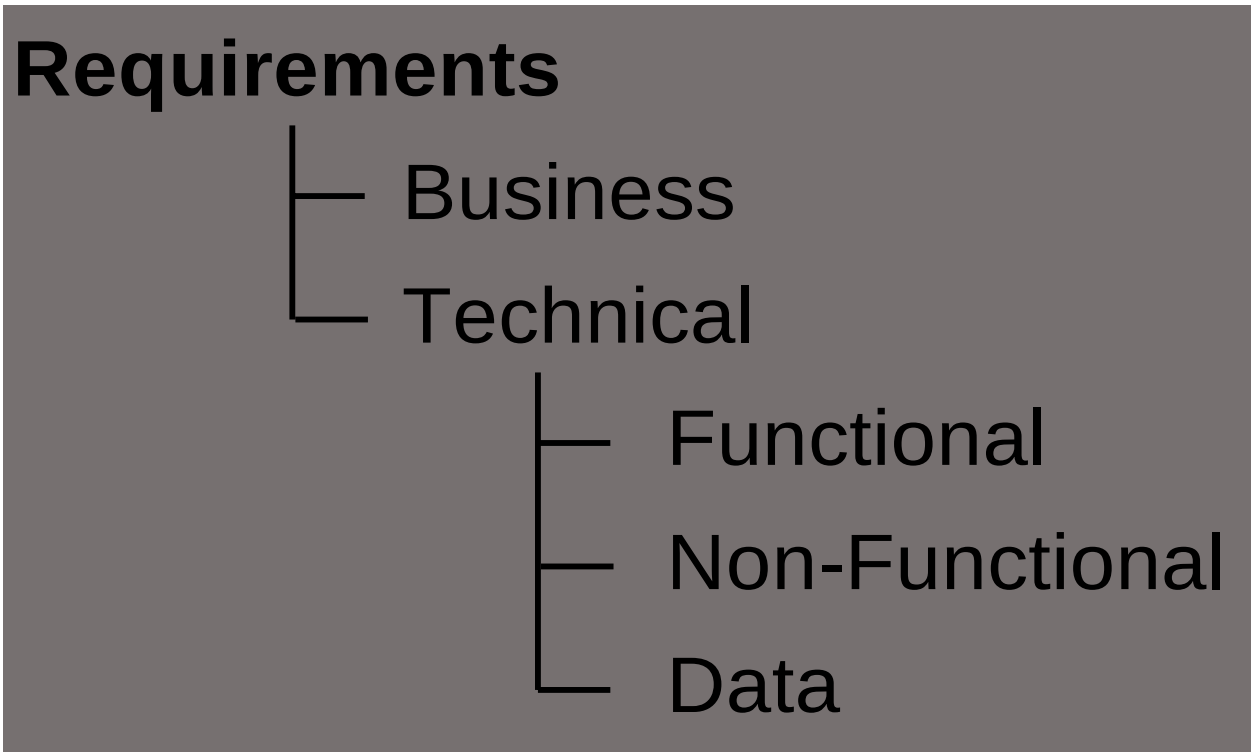


Fig 1: Waterfall Life Cycle Model

Types of Requirements:



Business Requirements:

- Business requirements relate to a business' objectives, vision and goals and are often captured by business analysts who analyze business activities and processes.
- **Example:**

If a company's need is to track its field employees by means of an employee tracking system, the business requirements for the project might be described as:

"Implement a web and mobile based employee tracking system that tracks field employees and increases efficiency by means of monitoring employee activity, absenteeism and productivity. "

Functional Requirements:

- A **functional requirement** is a statement of how a software product must map program inputs to program outputs.
- Functional requirements are specifications of the product's externally observable behavior, so they are often called **behavioral requirements**.

Examples:

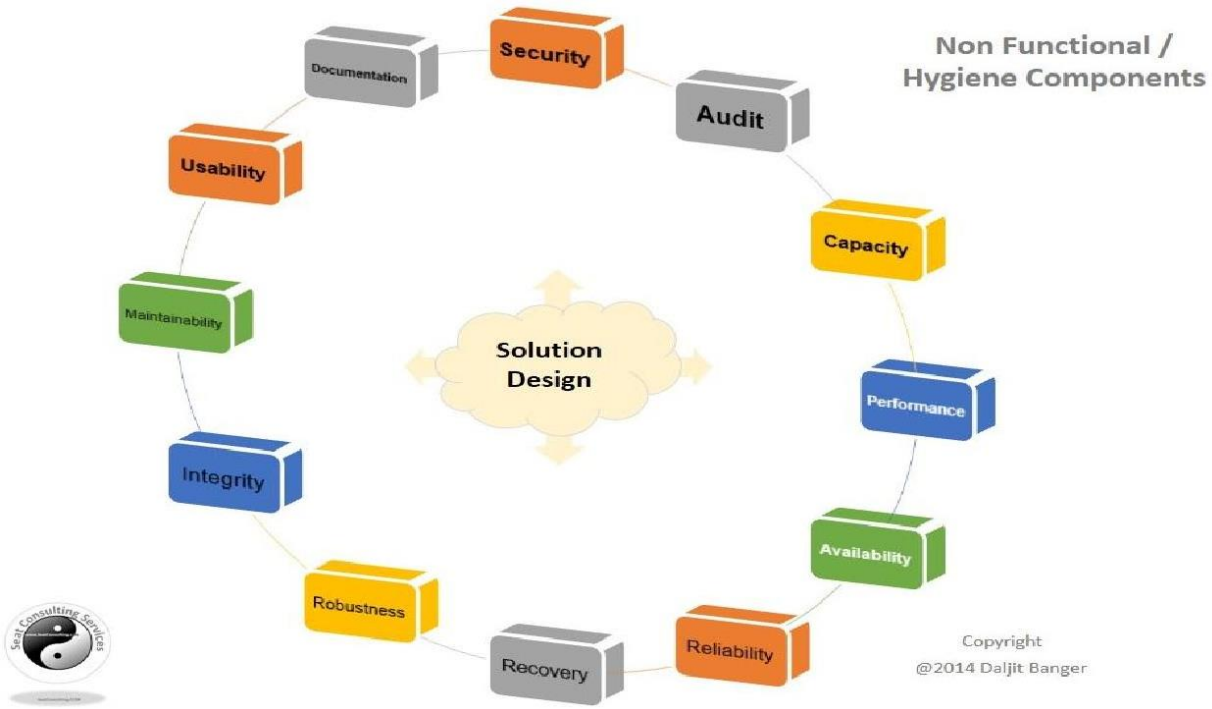
- Display the name, total size, available space and format of a flash drive connected to the USB port.
- Upon request from managers, the system must produce daily, weekly, monthly, quarterly, or yearly sales reports in HTML format.

Non-Functional Requirements:

- A **non-functional requirement** is a statement that a software product must have certain properties.
- Non-functional requirements are also called **non-behavioral requirements**.

The following statements are examples of non-functional requirements:

- The payroll system must process the payroll for all XYZ Corp employees in six hours or less.
- The system must run without failure for at least 24 hours after being restarted, under normal conditions of use.



Data Requirements:

- A **data requirement** is a statement that certain data must be input to, output from, or stored by a product.
- Data requirements describe the format, structure, type, and allowable values of data entering, leaving, or stored by the product.
- The Computer Assignment System must store customer names in fields recording first, last, and middle names.
- The system must display all times in time fields with the format *hh:mm:ss* where *hh* is a two-digit military time hour field, *mm* is a two-digit military time minutes field, and *ss* is a two-digit military time seconds field.

Levels of Abstraction:

- A **user-level requirement** is statement about how a product must support stakeholders in achieving their goals or tasks.
- An **operational-level requirement** is a statement about inputs, outputs, operations, characteristics, etc. that a product must provide, without reference to physical realization.
- A **physical-level requirement** is a statement about the physical form of a product, its physical interfaces, or its data formats.

SRS Templates:

- There is no universal SRS template. There are many templates available in books and on the internet. Most employ requirements types and levels of abstraction to help organize the material.
- Templates must be adapted for the product at hand.

IEEE Template:

1. Product Description

1.1 Product Vision

1.2 Business Requirements

1.3 Users and Other Stakeholders

1.4 Project Scope

1.5 Assumptions

1.6 Constraints

2. Functional Requirements

3. Data Requirements

4. Non-Functional Requirements

5. Interface Requirements

5.1 User Interfaces

5.2 Hardware Interfaces

5.3 Software Interface

IEEE SRS Template;

1. Product Description

1.1 Product Vision

- A **product vision statement** is a general description of a product's purpose and form.

1.2 Business Requirements

- **Business requirements** are statements of client or development organization goals that a product must meet

1.3 Users and Other Stakeholders

- **A stakeholder** is anyone affected by a product or involved in or influencing its development. Developers must know who the stakeholders are so that they are all consulted (or at least considered) in designing, building, deploying, and supporting the product.

1.4 Project Scope

- The **project scope** is the work to be done in the project.

1.5 Assumptions

- An **assumption** is something that the developers may take for granted. It is important to make assumptions explicit so that all stakeholders are aware of them and can call them into question.

1.6 Constraints

- A **constraint** is any factor that limits developers. Of course developers must be aware of all constraints

2. Functional Requirements

- **Functional requirements** are specifications of the product's externally observable behavior.

IEEE Format:

3. Data Requirements

- **Data requirements** describe the format, structure, type, and allowable values of data entering, leaving, or stored by the product.

4. Non-Functional Requirements

- A non-functional requirement is a statement that a software product must have certain properties.

5. Interface Requirements

5.1 User Interfaces

5.2 Hardware Interfaces

5.3 Software Interfaces

SRS Description:

- In this template, the design problem is documented in the "Product Description" section, which contains most of the information from the project mission statement. If a project mission statement exists, it should be referenced rather than reproduced...
- The sections named "Functional Requirements," "Data Requirements," and "Non-Functional Requirements" contain specifications mainly at the user and operational levels of abstraction.

SRS:

- The SRS is often referred to as the “parent” document because all subsequent project management documents, such as design specifications, statements of work, software architecture specifications, testing and validation plans, and documentation plans, are related to it.
- It’s important to note that an SRS contains functional and nonfunctional requirements only; it doesn’t offer design suggestions, possible solutions to technology or business issues, or any other information other than what the development team understands the customer’s system requirements to be.
- **Example:** https://www.scribd.com/doc/11934168/SRS-of-ATM

(WEEK 8 COMPLETED)