Unit 5: Documentation

AGENDA

- Documentation
  - System, Design, Operational, User,
- Time Chart, Budget Chart.

Documentation

- Documentation is considered to be good if it has the following qualities:

a) **Availability**: It should be accessible to those for whom it is intended.

b) **Objectivity**: It must be clearly defined in a language that is easily understood.

c) **Cross-referable**: It should be possible to refer to other documents.

d) **Easy to maintain**: When the system gets modified, it should be easy to update the documentation.

e) **Completeness**: It should contain everything needed, so that those who have gone through it carefully can understand the system.

Documentation

- There are five major types of documentation. They are:
  - (i) System Documentation
  - (ii) Design Documentation
  - (iii) Operational Documentation
  - (iv) User Documentation

System Documentation

- Each phase in the systems development cycle is accompanied by appropriate documentation, even if it is initially mark verbally, eventually must be written.
- It is desirable for the client and a systems analyst to work jointly in writing the request since each can contribute knowledge the other does not have.
- The written system's request is merely a statement of the user's problem.
- In documenting the results of its deliberations, the selection committee must specify the following:
- (a) The objectives of the impending feasibility study
- (b) The extent of the authority of the feasibility team
- (c) The individual or group responsible for completing the study

System Documentation

- **A feasibility report** is probably the most important form of documentation in the system development cycle. It accomplished the following two purposes:
- (i) It defines the objectives of the proposed system's change in reasonable detail after a sufficiently detailed study.
- (ii) It gives a plan to attain these objectives.

System Documentation

- At various points during systems design, the designing team produces the following additional forms of documentation:
- (i) File Specification: detailed definitions of each file in the system, best done in graphic form.
- (ii) Transaction Specification: detailed descriptions of all output anticipated from the system.
- (iii) Output Specifications: detailed descriptions of all output anticipated from the system.

System Documentation

- Documentation also includes plans to test the system and convert from the old to the new one.
- The systems analyst must also provide a plan to train the personnel affected by the changes.
- During the life cycle of the completed system, the system itself must provide documentation of how well it is operating and consequently should be designed to yield data about itself as a normal by-product.

Design Documentation

- **Design Document**
- Design documents are created to coordinate efforts of a large team, give them a stable reference point, and describe all parts of the software and how they will operate.
- Software Design Document is a written document that provides a description of a software product in terms of architecture of software with various components with specified functionality.
- The design specification addresses different aspects of the design model and is completed as the designer refines his representation of the software.
- These design documents are written by software engineers/designers or project managers and further passed to the software development team to give them an overview of what needs to be built and how.

Design Documentation

- Software design documents not only help others understand your system and provide documentation for future projects, but it also forces you to think through the entire system architecture.
- **What You Should Include in Your Software Design Document?**
- A typical software requirements document should involve the following details:
- **Title:** Add the title of the software design document.
- **Introduction:** Provide an overview of the entire document.
- **System Overview:** Provide a general description and functionality of the software system.
- **Architectural Strategies:** Describe the strategies that will be used that will affect the system.

What You Should Include...

- **Design Considerations:** Describe the issues that need to be addressed before creating a design solution:
  - **Assumptions and Dependencies:** Describe any assumptions that may be wrong or any dependencies on other things.
  - **General Constraints:** Describe any constraints that could have an impact on the design of the software.
  - **Goals and Guidelines:** Describe any goals and guidelines for the design of the software.
  - **Development Methods:** Describe the software design method that will be used.

What You Should Include...

- **System Architecture:** This section should provide a high-level overview of how the functionality and responsibilities of the system were partitioned and then assigned to subsystems or components.
- **Policies and Tactics:** Describe any design policies and/or tactics that do not have sweeping architectural implications (meaning they would not significantly affect the overall organization of the system and its high-level structures).
- **Detailed System Design:** Most components described in the system architecture section will require a more detailed discussion. Other lower-level components and subcomponents may need to be described as well.
- **Glossary:** An ordered list of defined terms and concepts used throughout the document.

Operational Documentation

- A well designed system may run for a long time with little or no assistance from the systems department.
- This can be possible only when the system has been documented in a proper way.
- For smooth running of the system, the system operator must have complete knowledge about the job.
- The job instructions must be in a form readily accessible to the system operator and written in simple and understandable style.

Operational Documentation

- The run book is a collection of operator instructions for each program at an installation and typically contains:
- (i) Narrative, describing the job run
- (ii) Listing of the programmed error conditions
- (iii) Detailed information for running the job, including:
  - input/output forms to be used
  - anticipated problem areas and how to handle them
  - detailed description of file assignment of each input/output device
  - disposition of data files after completing the job
  - general block diagram of the programming logic
  - restart procedures

User Documentation

- Systems users require proper documentation to prepare a developing system and to smoothly carry out existing ones.
- To meet this requirement, each system should have a manual that spells everything the users must know to do their job correctly.
- Users require two general types of information; complete details to handle each case the system processes, and overall picture of the system so that they can see their role in the total operation of the company.

User Documentation

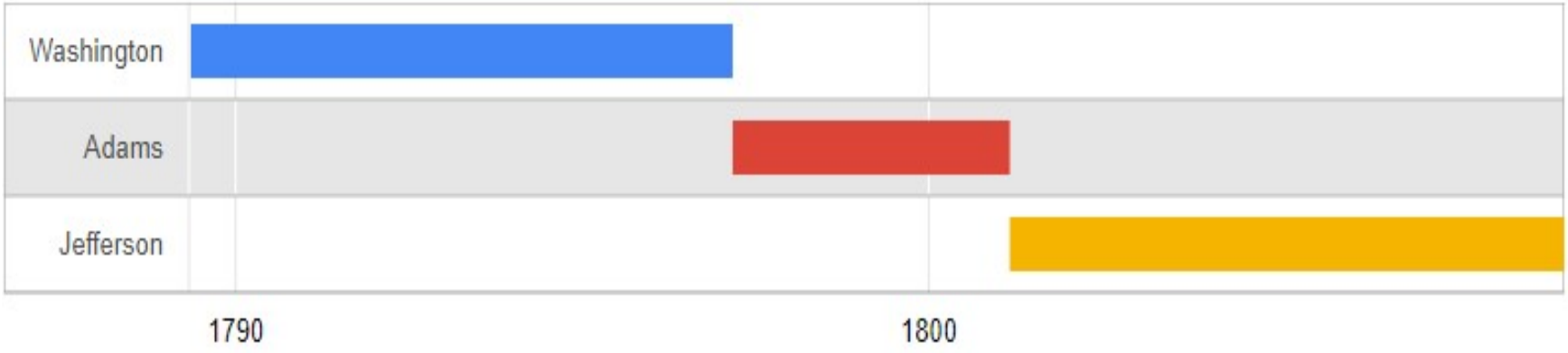
- The manual should supply the following information.
  - General flowchart of the system
  - Assignment of responsibility for specific tasks
  - Standards for work flow, including target dates and deadlines for specific tasks
  - Simple input and output documents
  - Detail procedures
  - Anticipated exceptions and instructions on how to handle them
  - Accuracy standards for data in the system

User Documentation

- A staff member in the user department must have an authority to consult when faced with a case not handled before.
- Properly prepared manual which is always available can provide the information needed by user.
- Supervising staff in user areas must understand the overall picture in each system just as staff members must understand the details of their function.
- This requires documentation, in the form of charts, graphs and illustrations, so that the supervising staff has a clear grasp of their department's role in the total system.

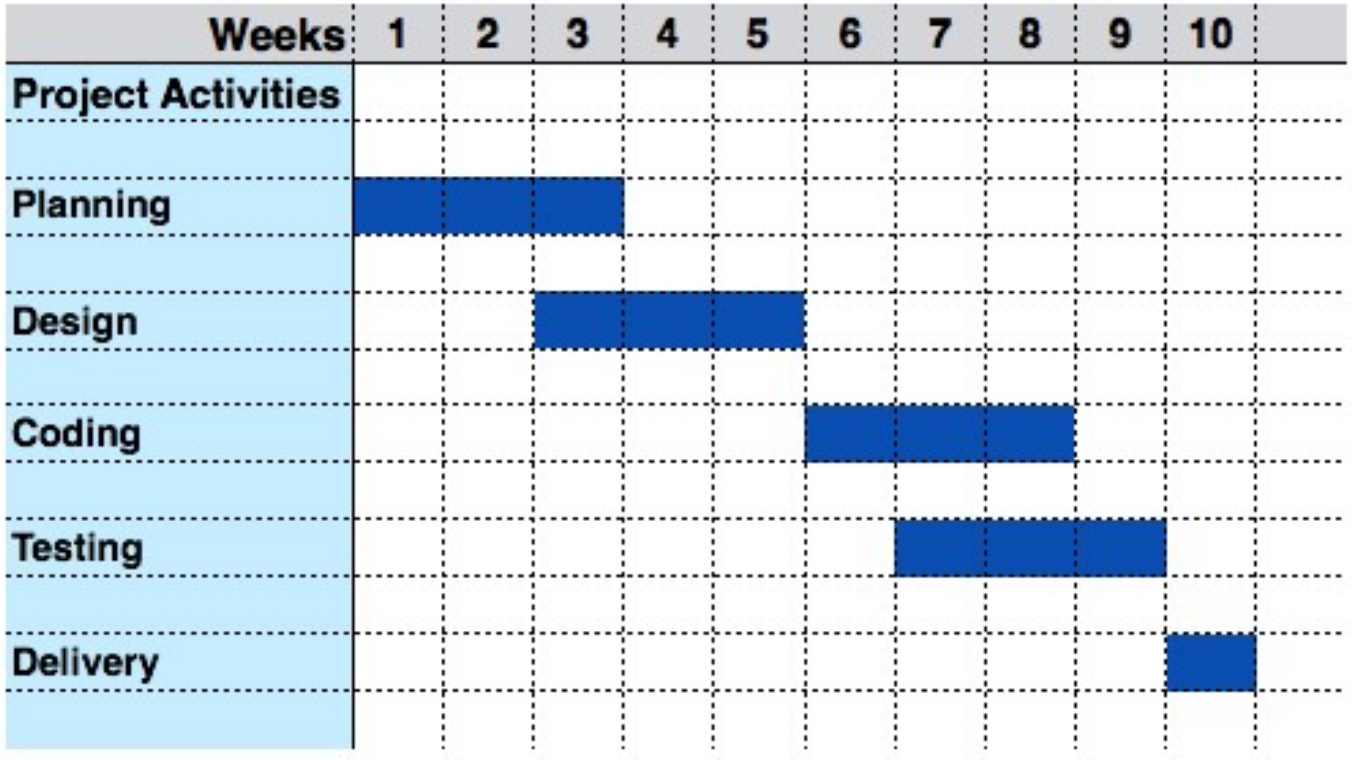
Time Chart

- Timeline charts aim to describe the chronological order of past and future events on a time scale.
- This type of chart is often used to oversee team workload and understand the overall progress of a specific project.
- Besides, timeline charts highlight specific events and milestones and help you visualize who's working on what and when.
- **A simple example**
- Let's say you want to plot when American presidents served their terms. Here, the "resources" are the presidents, and we can plot each president's term as a bar:



Time Chart

- A timeline chart is a visual representation of a series of events and can describe literally everything that happened or will happen in a given period of time in software development.



Budget Chart

- Managing budgets for software development projects is a challenging feat. The traditional approaches used for years are becoming increasingly obsolete.
- This makes it difficult to accurately plan production monitoring budget and all the other essential pieces required to build an effective application that meets customer demands.
- **Benefits of Effective Budgeting in Software Projects**
- Effective budgeting plays a pivotal role in the success of projects.
- Through careful planning and strategic allocation of resources, effective budgeting ensures financial stability and enhances overall project efficiency.
- Here are some of the benefits of a well-planned development budget:

Budget Chart

- **Streamlined Resource Allocation.** Effective software development budgeting encourages meticulous analysis of project requirements, enabling teams to allocate resources optimally.
- By identifying the key components of the project and estimating their costs accurately, project managers can prevent overspending, minimize wastage, and ensure that resources are dedicated where they are most needed.
- **Timely Project Completion.** Clear financial guidelines help avoid delays caused by sudden resource shortages or unexpected expenses.
- With a comprehensive software development budget in place, teams can plan their activities efficiently, reducing the likelihood of setbacks and ensuring that the project is completed on time, if not ahead of schedule.

Budget Chart

- **Risk Mitigation.** By setting aside contingencies, project managers are better equipped to handle unexpected events without jeopardizing the project's financial stability.
- **Stakeholder Communication.** Clear budgetary guidelines enable effective discussions about project priorities, goals, and potential trade-offs.
- **Financial Transparency.** By documenting all expenditures and tracking progress against the budget, teams can identify patterns, inefficiencies, and areas for improvement.

Budget Chart

- **Key Factors Influencing Your Software Development Budget**
- Several critical factors can significantly impact the overall cost of a software project and by understanding these key variables, you can create a more accurate and realistic budget that ensures the success of the project.

1. **Project Scope and Complexity**

- The scope of your software project refers to the features, functionalities, and requirements that your software will encompass.
- A more extensive and complex scope often translates to a higher budget.
- For instance, building a simple mobile app will generally cost less than developing a complex enterprise-level software solution with numerous integrated modules.

Budget Chart

2. Technology Stack and Platform

- The choice of technology stack and platform can significantly impact your budget.
- Different technologies have varying levels of complexity, scalability, and licensing costs.
- For example, opting for open-source technologies can reduce licensing expenses, while choosing proprietary software might lead to higher initial costs.

Budget Chart

3. Development Team Composition

- The size and composition of your development team play a crucial role in budget planning. A larger team with diverse skills will likely require a higher budget.
- Hiring experienced developers and specialists can also lead to increased costs compared to junior developers.
- In addition to developers, you should also consider the cost of other specialists:
  - Project managers (19-45$\per hour)
  - QA testers (28-35$\per hour)
  - UX\UI designers (25-39$\per hour)
  - DevOps specialists (61-73$\per hour)

Budget Chart

4. **Development Timeframe**

- The duration of your software development project directly affects your budget.
- Longer development timeframes often mean more resources, extended salaries for the team, and increased operational costs.
- Tighter deadlines might require additional resources to expedite development, potentially leading to higher expenses.

5. **In-House vs. Outsourcing**

- Deciding between in-house development and outsourcing can significantly impact your budget.
- While in-house development might provide more control, it can also involve higher personnel and infrastructure costs.
- Outsourcing, on the other hand, can offer cost savings by utilizing external expertise and resources.

Budget Chart

6. **Third-Party Integrations**

- If your software project requires integration with external services, APIs, or software solutions, it can influence your budget.
- Some third-party integrations might come with licensing fees or require additional development efforts, contributing to the overall cost.

7. **Regulatory and Compliance Requirements**

- Industries such as finance, healthcare, and government have strict regulatory and compliance standards.
- Meeting these requirements might involve additional development, security measures, and documentation, leading to increased costs.

Budget Chart

8. **Maintenance and Support**

- Post-launch maintenance and support are often overlooked but vital aspects of budget planning.
- Regular updates, bug fixes, and user support require ongoing resources and investment.

9. **Project Location and Cost of Living**

- The geographic location of your development team can influence costs.
- Developers in regions with a higher cost of living might demand higher salaries, impacting your budget allocation.

Budget Chart

- By carefully analyzing and addressing these key factors, you can develop a software development budget that aligns with your project's goals, scope, and requirements.
- For different software development models, budgeting strategies may be different.
- To calculate overall cost of project,
  - Identify key decisions,
  - Breakdown software features, and
  - Budget each high-level feature.

Budget Chart

- Sample

IT PROJECT COST BENEFIT ANALYSIS

COMPANY NAME				DATE CONDUCTED	XX/XX/XXXX
PROPOSED PRODUCT / INITIATIVE / SERVICE				COMPLETED BY	
CONSTANT OR CURRENT DOLLARS		STATUS QUO OR ALTERNATIVE		YEARS	XXXX - XXXX

SYSTEM LIFE COST PROFILE									
COST CATEGORY	YEAR 1	YEAR 2	YEAR 3	YEAR 4	YEAR 5	YEAR 6	YEAR 7	YEAR 8	TOTAL
Hardware									$ -
Servers									$ -
Desktop	$ 200,000								$ 200,000
Telecommunication Equipment									$ -
Software (Packaged or Custom)		$ 3,000							$ 3,000
Computer Room Upgrades				$ 2,500					$ 2,500
Furniture and Fixtures									$ -
Project Organizational/Support Costs									$ -
Planning (upon Approval)			$ 3,670						$ 3,670
Procurement									$ -
Contract Negotiations					$ 2,500				$ 2,500
Labor									$ -
Infrastructure						$ 600			$ 600
Development							$ 2,561		$ 2,561
Business Process Owners (Users)								$ 241	$ 241
Management									$ -
Training of Employees (Pre-Implementation)									$ -
Transition Costs (Parallel Systems)									$ -
Post Implementation Reviews									$ -
									$ -
									$ -
									$ -
TOTAL PROJECTED COSTS	$ 200,000	$ 3,000	$ 3,670	$ 2,500	$ 2,500	$ 600	$ 2,561	$ 241	$ 215,072
TOTAL PRESENT VALUE COSTS									$ -
CUMULATIVE TOTAL PROJECTED COSTS	$ 200,000	$ 203,000	$ 206,670	$ 209,170	$ 211,670	$ 212,270	$ 214,831	$ 215,072	

Budget Chart

- Sample

Addressing Agile Software Project Budget Assessment

This slide provides information regarding agile software project budget assessment in terms of project phases, estimated hours, total costs involved, etc.

Project Name	Project ABC	Project Number	12345678
Start Date	XYZ Ltd.	Start Date	20 Nov 2020
Project Manager	Martha Thatcher	End Date	10 April 2021
Remarks	Add text here	Total # Deliverable:	54

Project Phase		Estimated Hours	Developers		Analysts		Other Cost	Total Cost	Status
			Req.	Avg. Cost	Req.	Avg. Cost			
1	Phase –Design	12	3	$90	6	$90	$90	$900	
1.1	Sub Task 1	5	1	$30	1	$30	$30	$330	
1.2	Sub Task 2	5	1	$30	0	$30	$30	$180	
1.3	Sub Task 3	2	1	$30	5	$30	$30	$390	
2	Phase –Development	20	13	$90	6	$90	$90	$4,740	
2.1	Sub Task 1	5	1	$30	1	$30	$30	$330	
2.2	Sub Task 2	5	5	$30	0	$30	$30	$780	
2.3	Sub task 3	10	7	$30	5	$30	$30	$3,690	
3	Phase –Testing	6	7	$90	18	$150	$150	$1,020	
3.1	Sub Task 1	1	2	$30	3	$30	$30	$180	
3.2	Sub Task 2	1	2	$30	3	$30	$30	$180	
3.3	Sub Task 3	2	1	$30	3	$30	$30	$270	
3.4	Sub Task 4	1	1	$30	4	$30	$30	$180	
3.5	Sub Task 5	1	1	$30	5	$30	$30	$210	
	Total	38	23		30		$330	$6,660	

Estimated Hours	38
FTE	53
Total Cost	$6,660

Documentation

- **Program Documentation**
- Many companies discuss about programming documentation but fail to provide it adequately.
- Before a program is developed, the systems analysts should provide the programmer with the required documentation.
- The logic in some programs is best described by a flowchart. Sometimes, decision tables are most appropriate for explaining the logic of a program.
- Programmers should insist on proper documentation before starting a job. The programmer's responsibility in documentation is to provide information to enable future programmers to make necessary changes.

Program Documentation

- For continuity of information a company must insist on complete and meaningful documentation.

1. Four items constitute normal documentation required for each program.
2. Copying in final form of all input/output documents affecting the program.
3. Statement of standards for coding structures and input/output layouts.
4. Clarification of the program's interface with other related programs.
5. General flowchart or decision table.

Program Documentation

- Typically a documentation folder is provided for each program which contains all the **input/output forms** associated with the program, a detailed **flowchart** or **decision table** for the program use a set of operator and user instructions.
- Maintaining this type of documentation is costly and time consuming, for, programmers do not take interest in spending time for this type of work.
- Routine changes occur frequently in a program and all changes must be covered in the documentation folder.
- But the very changes which require the updating of existing documentation are the reasons for maintaining accurate documentation.