BACS2163: Introduction to Software Engineering

This note for subject Introduction to Software Engineering has been compiled and arranged by a salted fish.

In addition to the lecture slide content, the notes also include internet references or personal additions made by the compiler.

Please note that while every effort has been made to ensure accuracy, this note is intended for personal use and may not fully represent the entirety of the subject matter. Users are encouraged to consult primary sources and official course materials for comprehensive understanding and verification.

Table of Contents

🟩: midterm tips

BACS2163 - Introduction to Software Engineering

Midterm

c1

Types of system software/application software

Software Engineering (definition)

*Ethics no need

c2

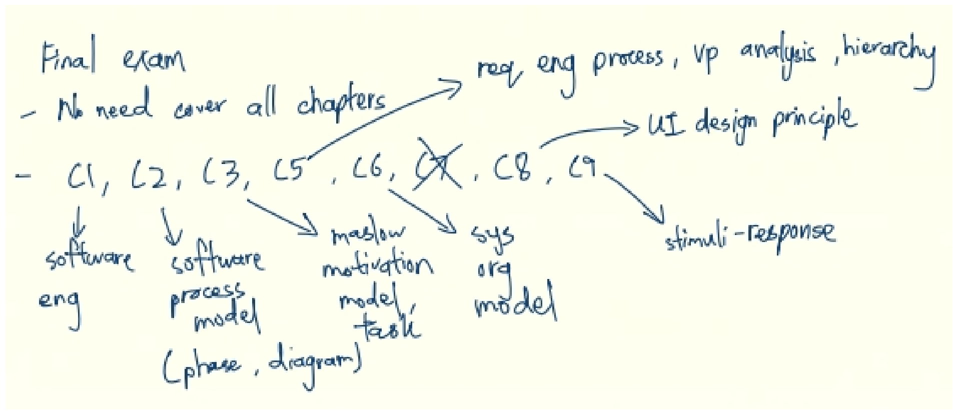
Software Process Model

c3

Gantt Chart

c4

Functional & Non-Functional Requirement



Final exam

- No need cover all chapters
- C1, C2, C3, C5, C6, ~~C7~~, C8, C9

software eng

software process model (phase, diagram)

maslow motivation model, task

req eng process, vp analysis, hierarchy

sys org model

UI design principle

stimuli-response

C1: software engineering

C2: spm(phase,diagram

C3: maslow, task

C5: req eng process, vp analysis+hierarchy

C6: sys org model

C8: UI design principle

C9: stimuli-response

C1: Introduction to Software Engineering

Software:

Software is a set of computer programs which are designed and developed to perform specific task desired by the user or by the computer itself.

Benefits

Automation and Efficiency: Automates tasks, increasing productivity.

Accuracy and Reliability: Reduces human error, ensuring consistent performance.

Scalability and Flexibility: Adapts to changing demands and requirements.

Cost Savings: Reduces operational costs through automation and optimization.

Continuous Improvement and Updates: Allows for regular updates to fix bugs and enhance functionality.

Functions

Manages computer resources

Provides tools for humans to utilize these resources.

Acts as an intermediary between human and stored info.

Characteristics

Software is developed or engineered, not manufactured in the classical sense.

It doesn't "wear out磨损" over time. Most software continues to be custom built to meet specific needs.

Types(2)...
System software...
Application software...

...System software

collection of programs designed to operate, control and extend the processing capabilities of the computer itself.

🟩**Types(3)**

System Control Programs -1
control the basic functioning of the computer, including the allocation of resources and management of system activities.

System Support Programs -2
provide support for various system operations, ensuring the smooth running of the computer.

System Development Programs -3
Tools used in the development, design, and testing of other software applications.

Examples(5)
Operating System -1
Utility Programs -2
Translators -3
Compilers -3
Loaders -3

...Application software

refers to the programs developed to perform specific tasks or provide services to end-users.

🟩**Types(2)...**

Generic...

Bespoke (or customized)...

	generic software product	**bespoke software product**
• Area • Description • Cost • Requirement spec. • Example	• Develop for public • Cheaper- share among buyer • Controlled by s/w org • MS office, • Window OS	• Develop for a specific user • Expensive- pay by one customer • Controlled by user/ cust • Public bank system, TARC UMT • Library sys

...Generic products

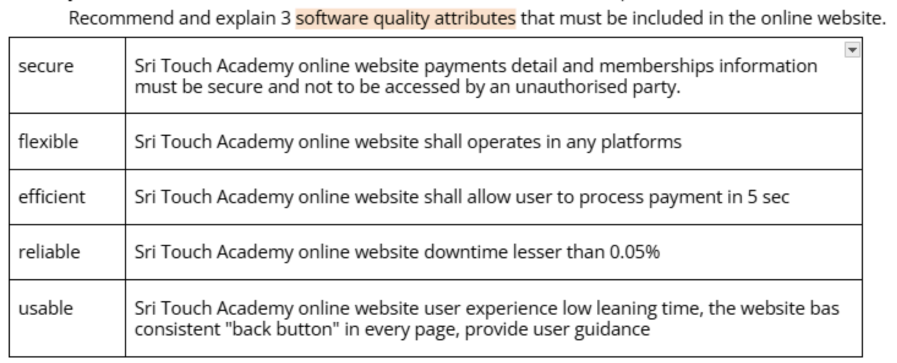
is designed to serve a wide range of customers with common needs, offering standardized solutions that can be used across various industries and applications.

Types of Generic Software:

Desktop Applications: Software designed to run on personal computers, providing functionality directly through the operating system's graphical user interface.

Web Applications: Software accessible through a web browser, allowing users to interact with applications hosted on remote servers via the internet.

Mobile Applications: Designed for smartphones and tablets, offering users the ability to perform tasks and access services on-the-go.



Recommend and explain 3 software quality attributes that must be included in the online website.

secure	Sri Touch Academy online website payments detail and memberships information must be secure and not to be accessed by an unauthorised party.
flexible	Sri Touch Academy online website shall operates in any platforms
efficient	Sri Touch Academy online website shall allow user to process payment in 5 sec
reliable	Sri Touch Academy online website downtime lesser than 0.05%
usable	Sri Touch Academy online website user experience low leaning time, the website bas consistent "back button" in every page, provide user guidance

...Bespoke (or customized) products

is created to meet the specific requirements of a particular client or organization. It is tailored to fit the unique business processes and goals of the user.

Attributes From a User's Perspective: (6)

Availability: The system is accessible and functions as needed during its operational period.

Functional: must offer precise features that meet the user's needs, as outlined in the question, and operate exactly as intended.

Efficient: Executes tasks with optimal use of time and resources.

Reliable: Delivers expected functionality without frequent failures.

Secure: Protects against unauthorized access and data loss.

Usable: satisfying user experience that is intuitive and easy to navigate.

From a Developer's Perspective: (5)

Flexible: Capable of being adapted to new uses or different environments.

Maintainable: The design allows for easy updates and fixes by maintenance programmers.

Reusable: Enables the reuse of software components in other projects.

Buildable: The design is not overly complex, and can implement it effectively.

Manageable: Facilitates easy estimation of work involved and monitoring of progress.

🟩🟦Software Engineering

Software engineering is an engineering discipline, which is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use

Importance of Software Engineering:

Software engineering aims for cost-effective development of software systems, ensuring projects are on schedule, reliable, within budget, and meet user needs.

Software Costs

The cost of software development includes not only the initial creation but also the ongoing maintenance and updates required to keep the software relevant and secure.

4 layers of software engineering: https://prnt.sc/nQ7r7S_NrT25

1. quality control(sqa)
2. process(Spiral..)
3. method(OOAD)
4. tools(Microsoft Project)

Key Challenges of Software Engineering

Legacy Challenge: Maintaining and updating existing software to avoid excessive costs while ensuring business continuity.

Heterogeneity/Diversity Challenge: Developing techniques for building dependable software flexible enough to handle various computer systems and support structures.

Delivery Challenge: Reducing delivery times for large and complex systems without compromising quality.

System Engineering

System engineering is an interdisciplinary field that addresses the design and management of complex engineering projects. It is composed of a variety of system elements that encompasses包括 **software engineering**.

Professional & Ethical Responsibilities of Software Engineers

Software Engineering ethics

Issues

Ethical issues in software engineering involve balancing the need for innovation and efficiency with the responsibility to protect user privacy, ensure data security, and maintain professional integrity.

ACM/IEEE Code of Ethics:

PUBLIC: Act in the public interest.

CLIENT AND EMPLOYER: Prioritize client and employer interests.

PRODUCT: Ensure product quality.

JUDGMENTMaintain professional integrity in judgment.

MANAGEMENT: Promote ethical management in software development.

PROFESSION: Advance the profession's integrity and reputation.

COLLEAGUES: Be fair and supportive of colleagues.

SELF: Engage in continuous professional development and ethical practice.

C2: Software Process Model 🟩

Process?, Characteristics, Important?

A process is a series of steps taken to produce an intended output	• Prescribes all major process activities • Uses resources, subject to a set of constraints (e.g., schedule, number of people) • Produces intermediate and final products • May be composed of sub-processes with hierarchy or links • Each process activity has entry and exit criteria • Activities are organized sequentially, ensuring clear timing • Each process has guiding principles, including goals of each activity	• Impose consistency and structure on a set of activities • Guide us to understand, control, examine and improve the activities • Capture experiences and pass them along

Software Process(4)

Software Specification	Detailing the software system's functional and non-functional requirements.	
Software Development	Designing, programming, documenting, testing, and bug fixing.	
Software Validation	Ensuring the software meets business requirements and user needs.	
Software Evolution	Timely updates for various reasons after initial development.	

🟦Software Process Models (phase and diagram)

is a simplified description of a software process, presented from a particular perspective.

SDLC Models

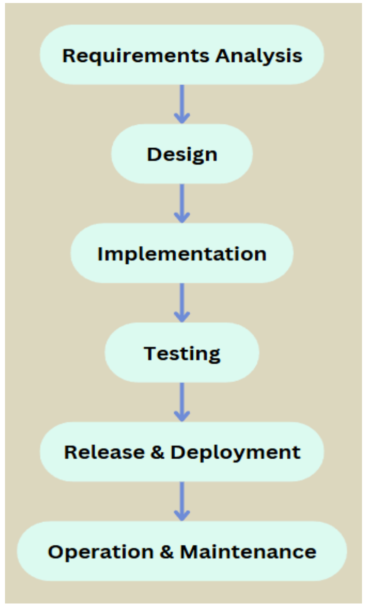
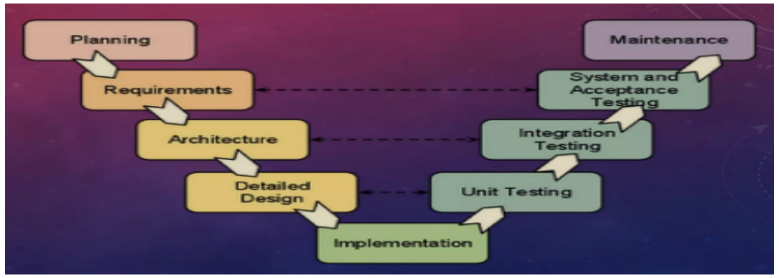
Includes:

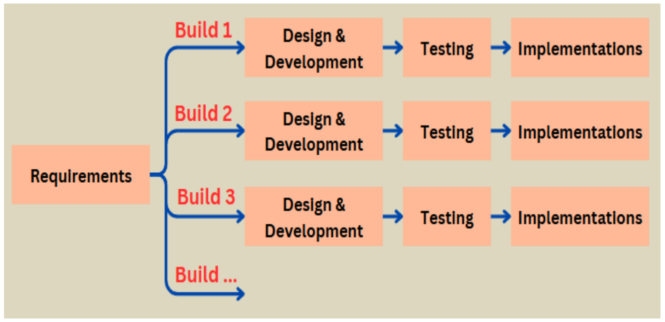
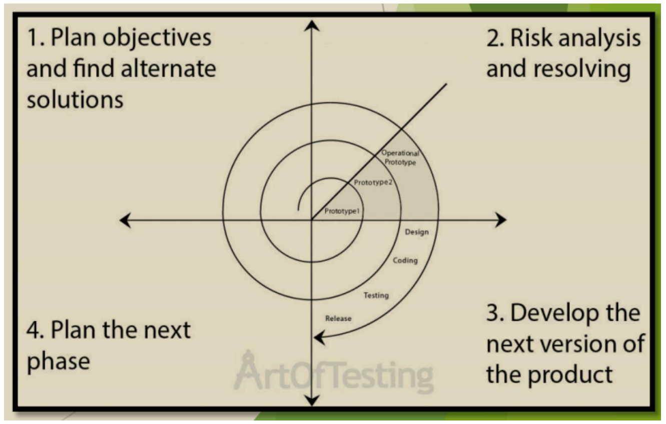
Adaptive Methodology	Predictive Methodology
Requirements and schedules are not known in advance and are managed in an agile and iterative fashion. Eg: Agile-driven approach.	Requirements and schedules are known in advance (e.g., Waterfall Model). Eg: plan-driven approach.

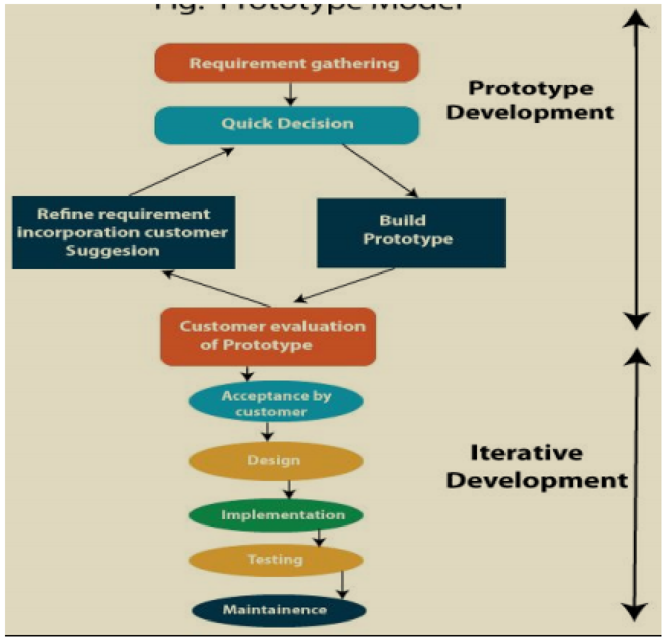
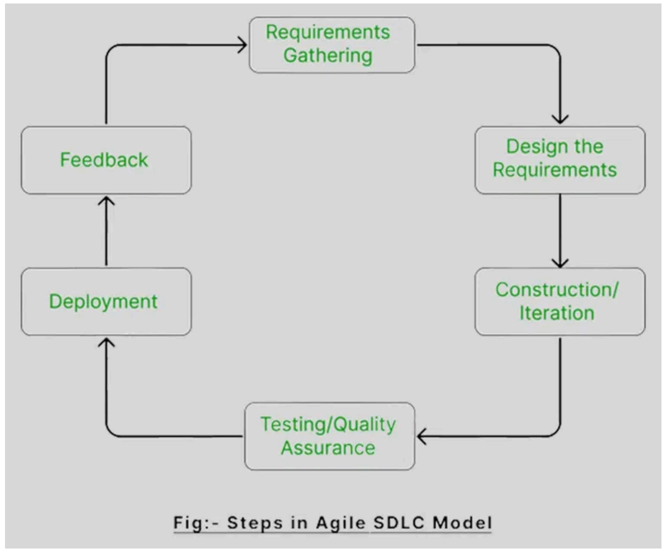
Agile and Plan-driven Methods

Agile Methodology...	Plan-Driven Methods
characterized by its flexibility, integration of activities, and responsiveness to change, developing requirements and design together.	follows a structured sequence of stages with a strong emphasis on documentation and predefined requirements, making it less adaptable but potentially more predictable

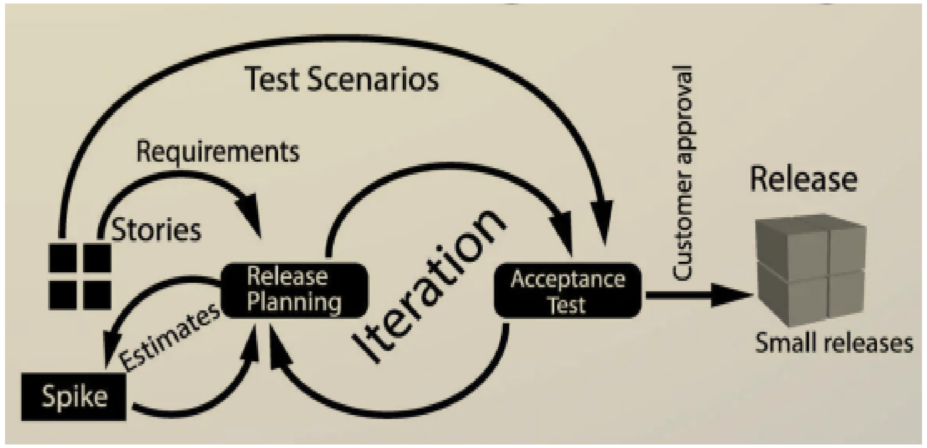
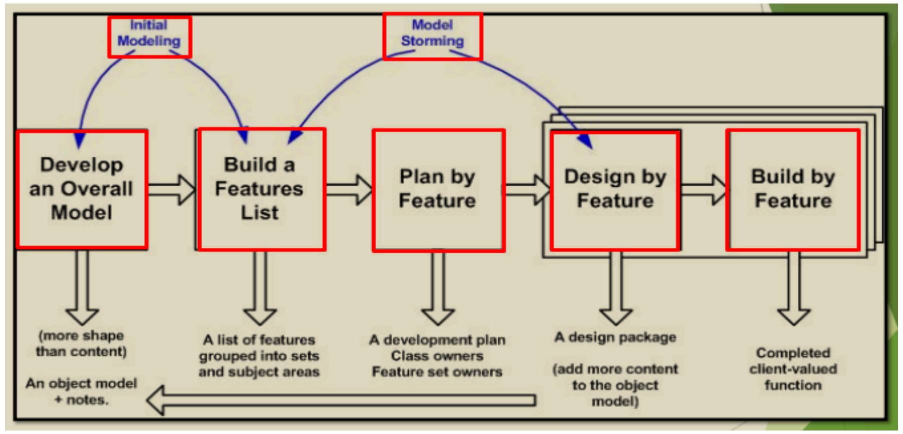
Detailed SDLC Models:

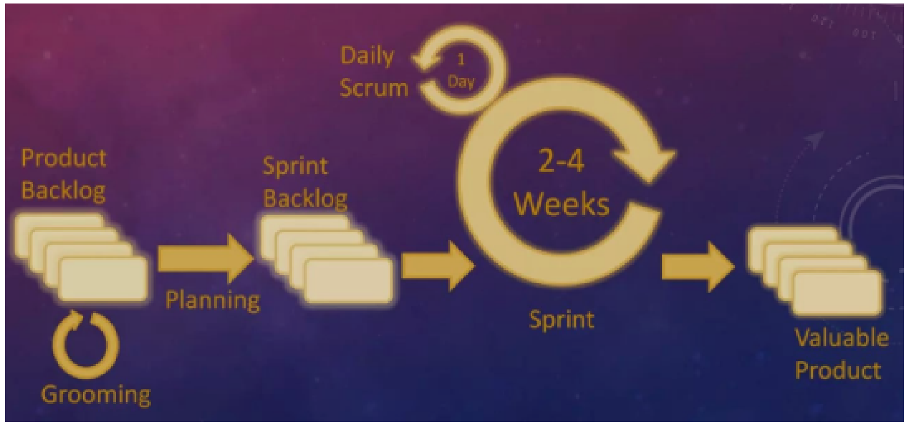
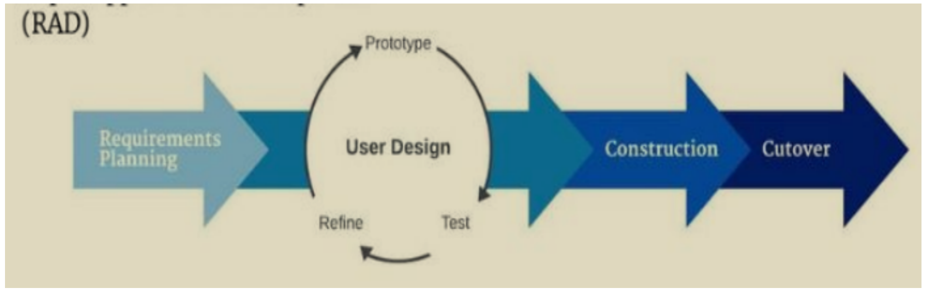
Waterfall  Requirements Analysis Design Implementation Testing Release & Deployment Operation & Maintenance	A linear sequential model where progress flows in distinct phases: ✅: Clear structure, easy to understand, suitable for fixed requirements. ❌: Inflexible, delayed feedback, not suitable for complex or high-risk projects.	• large project • fixed requirement as changes costly • not support iteration
	V-shaped Model (extension of Waterfall) with increased emphasis on verification and validation at each phase. ✅: Ensures thorough testing and verification. ❌: Can be time-consuming and is still not adaptive to change.  Planning Requirements Architecture Detailed Design Implementation Unit Testing Integration Testing System and Acceptance Testing Maintenance	• early testing phases • some iteration is acceptable but remains limited
Incremental	Requirements are divided into multiple standalone modules, each going through its own development lifecycle. Phases: Requirement, Design, Testing and Implementation ✅: Faster delivery of working software, flexible, easier to manage risk. ❌: Requires good planning and a clear definition of the whole system. Explain the process involved in incremental software process model. analysis and requirement (need very clear), repeat in version(1,2,3...): design,	• clear req, short staff, build by version • User can use the core function first • iteration

	implementation, testing deliver the core product function(s)/feature(s) at the first increment/version deliver the supplementary product function(s) feature(s) in next increment(s) version(s).	 Requirements Build 1 Build 2 Build 3 Build ... Design & Development Testing Implementations
Spiral	Combines elements of the Waterfall Model with iterative cycles that include risk analysis. **Four phases: Planning, Design, Construct and Evaluation** ✅: High risk analysis, useful for large, mission-critical projects. ❌: Can be costly, requires particular expertise for risk analysis.	• useful dealing with high risks projects - whereby failure of system can cause loss of life/ huge amount of money • Requirement are unclear and complex • large project  1. Plan objectives and find alternate solutions 2. Risk analysis and resolving 3. Develop the next version of the product 4. Plan the next phase Operational Prototype Prototype2 Prototype1 Design Coding Testing Release ArtOfTesting
Prototyping	Involves creating a working prototype of the system, refining it based on feedback, and evolving it into the final product. ✅: Early customer involvement, easy accommodation of new requirements. ❌: Potential for requirement instability, difficulty in finalizing the prototype. **4 types** 1. Rapid Throwaway prototypes 2. Evolutionary prototype 3. Incremental prototype 4. Extreme prototype	• User does not have any computer background - difficulty in requirement elicitation • Highly iterative, requirement not fully defined (user unclear req)

		 Requirement gathering Quick Decision Refine requirement incorporation customer Suggesion Build Prototype Customer evaluation of Prototype Acceptance by customer Design Implementation Testing Maintainence Prototype Development Iterative Development
Agile	Based on 12 principles emphasizing individuals and interactions, working software, customer collaboration, and responding to change. ✅: Highly adaptive, promotes rapid delivery and continuous improvement. ❌: May require a high level of discipline and self-organization from the team. **<u>12 principles</u>** 1. Customer satisfaction through rapid delivery of useful software. 2. Welcomes changing requirements, even late in development. 3. Frequent delivery of working software. 4. Working software as the principal measure of progress. 5. Sustainable development, maintaining a constant pace. 6. Close, daily co-operation between business people and developers. 7. Face-to-face conversation as the best form of communication. 8. Projects built around motivated individuals, who should be trusted. 9. Continuous attention to technical excellence and good design. 10. Simplicity—maximizing the amount of work not done. 11. Self-organizing teams. 12. Regular adaptation to changing circumstances.	• rapid and highly iterative • large projects with high costs  Requirements Gathering Design the Requirements Construction/ Iteration Testing/Quality Assurance Deployment Feedback Fig:- Steps in Agile SDLC Model https://www.geeksforgeeks.org/software-engineering-agile-development-models/

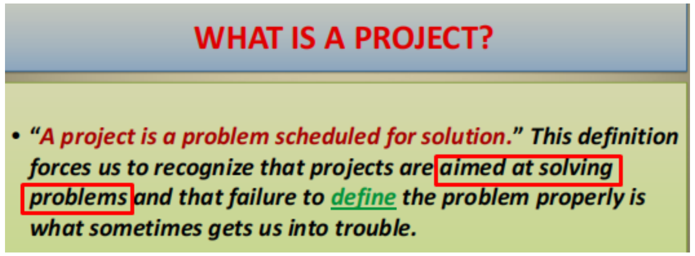
...Agile Methodology (4)

Extreme programming (XP)  Test Scenarios Requirements Stories Release Planning Iteration Acceptance Test Customer approval Release Small releases Estimates Spike	Focuses on extreme testing and customer involvement.	
Feature Driven Development (FDD)  Initial Modeling Model Storming Develop an Overall Model Build a Features List Plan by Feature Design by Feature Build by Feature (more shape than content) An object model + notes. A list of features grouped into sets and subject areas A development plan Class owners Feature set owners A design package (add more content to the object model) Completed client-valued function	Prioritizes delivering features in a short timeframe.	
SCRUM	Empowers teams to work in sprints with clear goals and roles.	

 Daily Scrum 1 Day Product Backlog Sprint Backlog 2-4 Weeks Planning Sprint Grooming Valuable Product		
RAPID APPLICATION DEVELOPMENT (RAD)  (RAD) Prototype Requirements Planning User Design Construction Cutover Refine Test	Focuses on quick development cycles with iterative feedback.	Pros,example when suitable to use? Cons,example when it is not suitable to use?

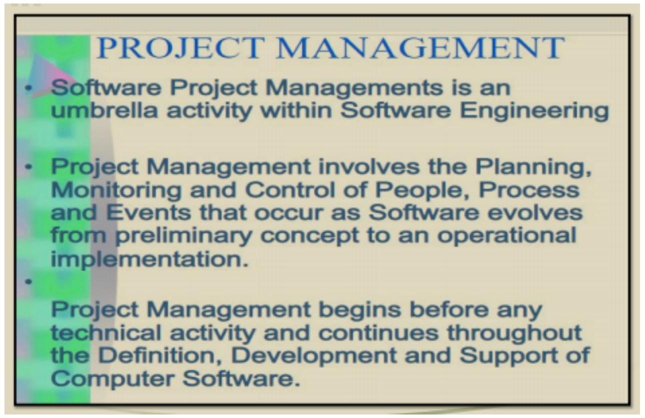
C3: Project Management

Project? Project Management? Importance?



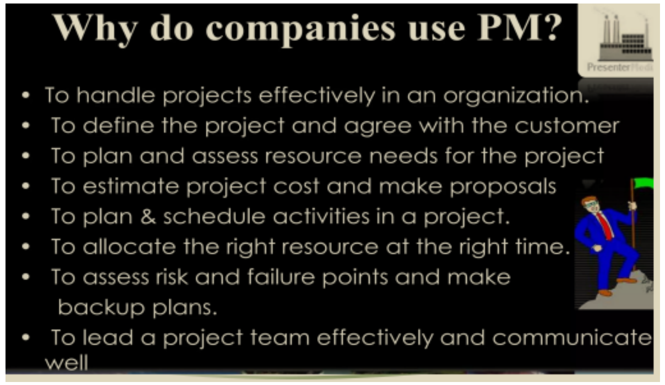
WHAT IS A PROJECT?

- "A project is a problem scheduled for solution." This definition forces us to recognize that projects are aimed at solving problems and that failure to define the problem properly is what sometimes gets us into trouble.



PROJECT MANAGEMENT

- Software Project Managements is an umbrella activity within Software Engineering
- Project Management involves the Planning, Monitoring and Control of People, Process and Events that occur as Software evolves from preliminary concept to an operational implementation.
- Project Management begins before any technical activity and continues throughout the Definition, Development and Support of Computer Software.



Why do companies use PM?

- To handle projects effectively in an organization.
- To define the project and agree with the customer
- To plan and assess resource needs for the project
- To estimate project cost and make proposals
- To plan & schedule activities in a project.
- To allocate the right resource at the right time.
- To assess risk and failure points and make backup plans.
- To lead a project team effectively and communicate well

🟦Tasks of Software Project Management

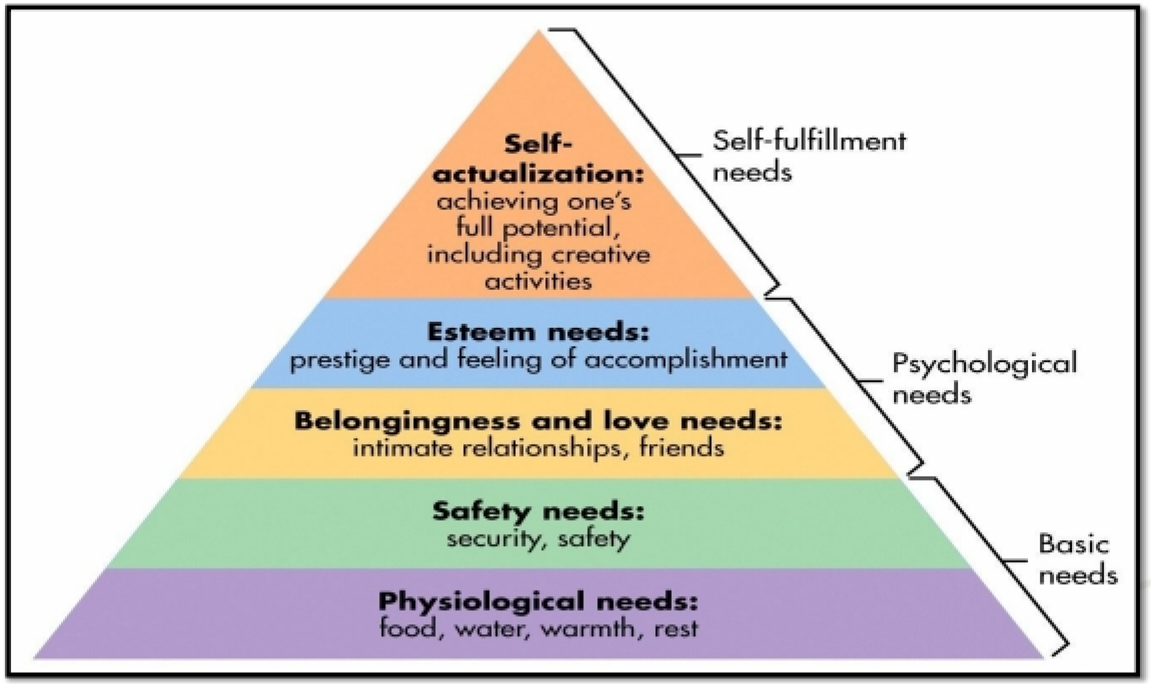
Main responsibilities of IT project manager

- Planning and scheduling project development
- Supervise the work to ensure it meets the required standards..
- Monitor progress to ensure it stays on time and within budget.

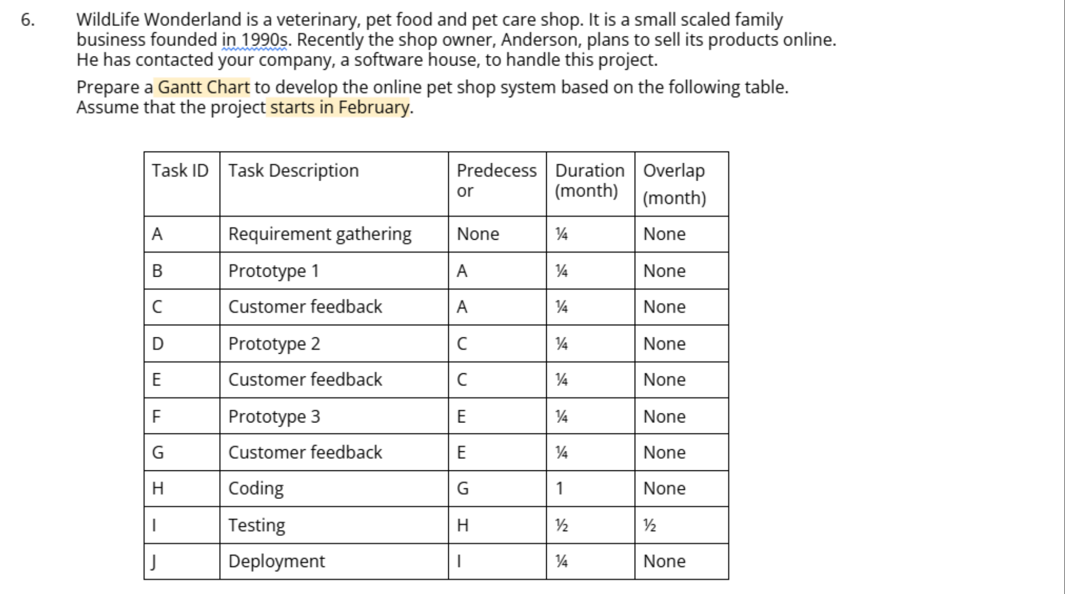
Management Activities(6)

Proposal writing	Describe objectives of the projects and how it will be carried out Usually includes cost and schedule estimations.
Project planning	Concerned with identifying the activities, milestones and deliverables. Example ● Requirement Specification ● Design Specification ● Prototype A project plan will be created to guide the development towards the goals Project Scheduling

	Involves breaking the work into tasks and estimating the time needed to complete each one. Usually represent as a chart showing ● Work breakdown ● Activities dependencies ● Staff allocation ● Graphically notation ● Gantt Chart ● PERT chart / CPM chart(Program Evaluation Review Technique / Critical Path method)
Project costing	Cost estimation: To estimate resource required to reach the project plan Example: ● Hardware ● Software ● People Cost-benefit analysis technique ● Payback period ● Return on Investment (ROI) ● Net Present Value (NPV)
Project monitoring	Is a continuous project activity Project managers need to keep track progress (monitor time, cost and quality) and compare actual progress against the plan. Common problem occur Behind schedule - Delay in task Over budget - Costs exceeding the plan Quality issues - Not meeting the agreed standards. Corrective action : Eg.: Add more staff, reassign task, using OT Formal/ Informal Reviews
Personnel selection and	Skilled staff with appropriate experience Project manager select right candidates join their team

evaluation 🟦	🟦**Maslow motivation model**  Self-actualization: achieving one's full potential, including creative activities Esteem needs: prestige and feeling of accomplishment Belongingness and love needs: intimate relationships, friends Safety needs: security, safety Physiological needs: food, water, warmth, rest Self-fulfillment needs Psychological needs Basic needs 🟩**Factor Influence Group Working** Group Composition: Right balance skills, experience and personalities in teams Group Cohesiveness: Group think itself is a "team" rather than collection of individuals Group Communication: Group member communicate effectively with each other Group Organization: Team members feel valued and satisfied with their role in the group.
Report writing and presentation	Project manager should write a clear and concise report. Strong oral and written communication skills are essential for them to communicate effectively.

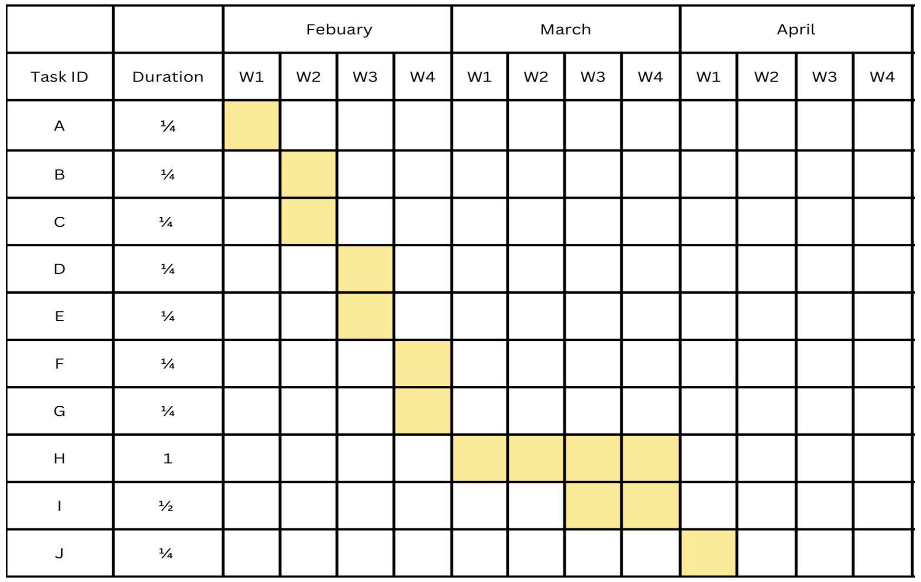
🟩Gantt Chart Example:



6. WildLife Wonderland is a veterinary, pet food and pet care shop. It is a small scaled family business founded in 1990s. Recently the shop owner, Anderson, plans to sell its products online. He has contacted your company, a software house, to handle this project.

Prepare a Gantt Chart to develop the online pet shop system based on the following table. Assume that the project starts in February.

Task ID	Task Description	Predecessor	Duration (month)	Overlap (month)
A	Requirement gathering	None	¼	None
B	Prototype 1	A	¼	None
C	Customer feedback	A	¼	None
D	Prototype 2	C	¼	None
E	Customer feedback	C	¼	None
F	Prototype 3	E	¼	None
G	Customer feedback	E	¼	None
H	Coding	G	1	None
I	Testing	H	½	½
J	Deployment	I	¼	None



		Febuary				March				April			
Task ID	Duration	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
A	¼	🟨											
B	¼		🟨										
C	¼		🟨										
D	¼			🟨									
E	¼			🟨									
F	¼				🟨								
G	¼				🟨								
H	1					🟨	🟨	🟨	🟨				
I	½							🟨	🟨				
J	¼									🟨			

第一行主时间第二行是跟题目分的部分时间；第一二列写taskid 和duration；Predecessor是表示在谁后面；Overlap表示在谁的后面前进多少；

C4: System Requirements

System requirements are more detailed and precise than user requirements, intended for different readers such as software developers, system architects, etc.

User Requirements	**System Requirements**
High-level, abstract statements for users and procurers. Use natural language, diagrams, and tables for easy understanding. Define what services the system should provide and under what constraints.	Precise, detailed descriptions; an extended version of user requirements. Use structured natural language, supported by system models and tables. Serve as the starting point for system design, also known as functional specifications. May form the basis of a contract between the developer and the customer.

🟩Functional vs. Non-Functional Requirements (Software System Requirements)

Functional Requirement	Non-Functional Requirements
• Services of system should provide, should not do in some case • How system react to particular input • How system behave in particular situation	Product • Requirements • Usability • Performance • Space • Reliability • Portability Process/Organizational Requirements • Delivery • Implementation • Standards External Requirements • Interoperability • Ethical

- Legislative
- Privacy
- Safety

Example

Penxoniks Sdn. Bhd. has recently launched its new products. In view of the sales that have been very encouraging since then, Mr. Alex Tham, the Sales Director of Penxoniks, has decided to computerize the sales and inventory system.

The software organization that you work with has successfully bid for the above mentioned project. You, as one of the project managers in your software organization, have been appointed to lead this project. During the initial requirement study exercise, your team noticed that the staff members of the Sales Department of Penxoniks are all very supportive of this upcoming project. They are able to explain their requirements very clearly in the recent interview sessions. Furthermore, most of them have highlighted to you that they hope the new system should have a very low learning time.

The top management of Penxoniks wants the new system, which will include an online order and payment feature, to be available before Christmas. They believe that this online feature will boost their sales during the Christmas

a. Compose the user and system requirements for the ***Online Order and Payment*** function mentioned in the scenario.

User requirement

1. The sales and inventory system includes online ordering and payment features to be developed in three months with using the Object-Oriented Analysis and Design Development Methodology and to be implemented based on the Oracle database. The new system's downtime should not be more than 0.05% and has very low learning time

System requirement spec

1. The system shall allow customer to place order online.
2. The system shall allow customer to cancel order online.
3. The system shall allow customer to modify order online.
4. The system shall allow customer to make a payment order online.
5. The system shall send order confirmation notification to customer email.
6. The system shall allow customer to track order online.
7. The system shall allow customer to view order history.
8. The system shall allow staff to generate online sales report.
9. The system shall allow staff to check the stock inventory online.

b. Non-Functional requirements can be categorized into three main categories, namely product, organisational and external.

Identify 4 non-functional requirements requested by the management and staff of Penxoniks in the above scenario. For each identified non-functional requirement, **indicate whether it is a product, organisational or external requirement**.

Non-Functional Requirement

Product
1.0 Reliability
1.1 The new system's downtime should not be more than 0.05%.

2.0 Usability
2.1 The new system has very low learning time, maintains consistency in every page, provides user guidance, and uses familiar icons.

and New Year festive season. Hence, the project team will only have about three months to work on this project. Top management has also emphasized that the new system's downtime should not be more than 0.05%. Furthermore, the new system should be developed using the Object-Oriented Analysis and Design Development Methodology and must be implemented based on the Oracle database.	3.0 Security 3.1 The system shall secure user's information like credit / debit card from unauthorized access. 4.0 Performance 4.1 The system shall process payment in 5 secs. Organization 1. The new system should be developed using the Object-Oriented Analysis and Design Development Methodology. 2. The new system should be implemented based on the Oracle database. 3. The system must develop in 3 months. External 1. The taxation shall conform to Malaysian Sales and Services Tax.

Natural Language Problems • Is over flexible Same thing can be said in many different ways Diagrams are better to scope/represent requirements • Lack of clarity: may cause ambiguity歧义? wordy啰嗦? difficult to read? • Requirement confusion: functional, non-functional requirements, system goals, design info may not be clearly distinguished • Amalgamation合并: several requirements expressed as a single requirement	**System Requirement Specification (SRS)** • Structured natural language (template table to specify.. • Design description language (PSL/PSA.. • Graphical notations (flowchart, sequence diagram.. • Mathematical specifications (Z-specification.. **Software Requirements Document** Not a design doc, is set out what system should do without specifying how it should done Users(5): 1. System customers 2. (project) manager 3. System engineers 4. System test engineers 5. System maintenance engineers Structures(10) 1. Preface 2. Introduction 3. Glossary

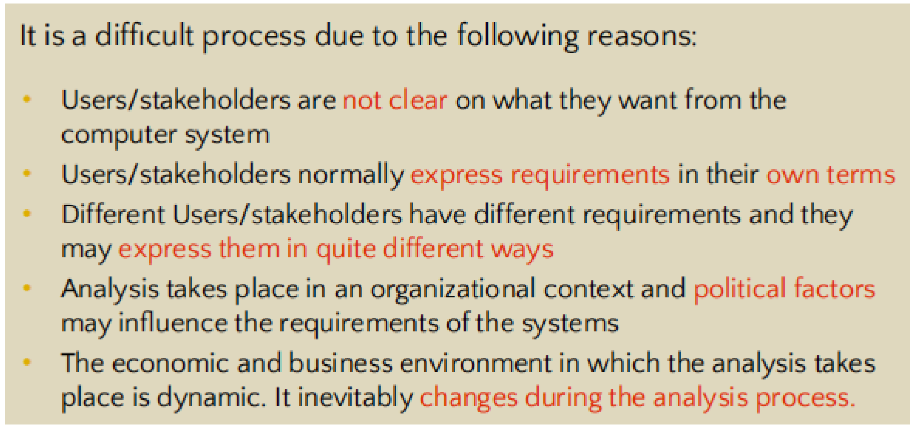
	4. User requirements definition 5. System architecture 6. System requirements specification 7. System models 8. System evolution 9. Appendices 10. Index

C5: Requirements Engineering Process

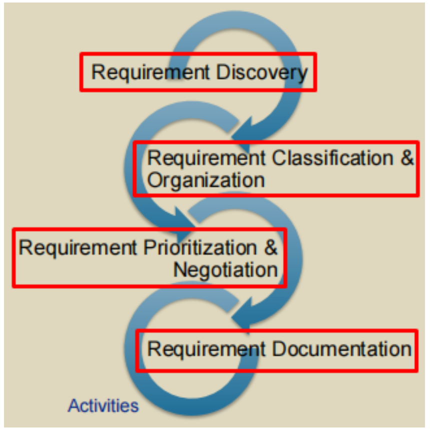
Requirement Engineering: process that involves all of the activities required to create and maintain a system requirements document (Requirements engineering is the process of identifying, eliciting, analyzing, specifying, validating, and managing the needs and expectations of stakeholders for a software system.)

🟦Requirement Engineering Process (5) 🟦hierarchy

1. Feasibility Studies (TELOS)	1. Technical feasibility: Is the project technically possible? 2. Economic feasibility: Does it make financial sense? 3. Legal feasibility: How can legal limitations affect the project? 4. Operational feasibility: How hard is it to maintain and manage the project? 5. Schedule feasibility: Can we keep up with realistic deadlines?
2. Requirement Elicitation引出 and Analysis	process of deriving得出 the system requirements. (Software engineers works with stakeholders)


It is a difficult process due to the following reasons:

- Users/stakeholders are not clear on what they want from the computer system
- Users/stakeholders normally express requirements in their own terms
- Different Users/stakeholders have different requirements and they may express them in quite different ways
- Analysis takes place in an organizational context and political factors may influence the requirements of the systems
- The economic and business environment in which the analysis takes place is dynamic. It inevitably changes during the analysis process.


Requirement Discovery
Requirement Classification & Organization
Requirement Prioritization & Negotiation
Requirement Documentation
Activities

Activities(4)

1. Requirement Discovery	🟦**Viewpoint-oriented Analysis**

Advantages	Disadvantages
• Multiple perspective Captures needs from different stakeholders, ensuring thorough coverage.	• Time-consuming Gathering and analyzing requirements from various viewpoints takes more time.

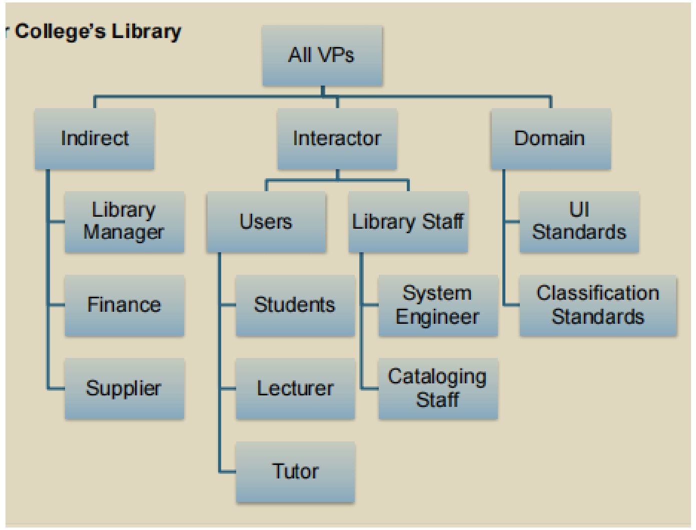
• Improve communication Helps developers and stakeholders communicate by focusing on specific viewpoints. • Conflict resolution Identifies and resolves conflicting requirements early by addressing different perspectives.	• Complexity Managing many viewpoints can lead to conflicting requirements, increasing complexity. • Integration difficulty Merging different viewpoints into a unified system can be challenging.

Viewpoints

Indirect Viewpoints: who do not use the system & Influence the requirements (High level requirements & constraints)

Interactor Viewpoints: Interact directly with system (System features & interfaces)

Domain Viewpoints: Provide domain constraints

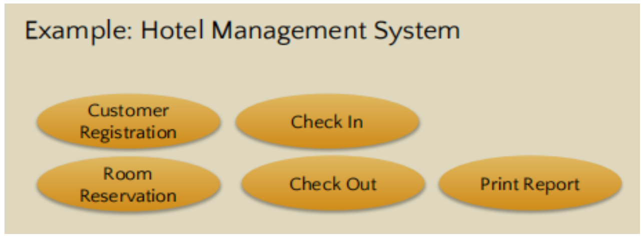


It is important to organize and structure the viewpoints into a hierarchy. WHY?

Answer: helps to identify viewpoints, ensure common requirements are shared without needing SE to gather each one separately, cover all VPs in the analysis and elicitation stage and to determine validity of VPs.

Techniques (4)

- Interviewing

	• Prototyping • Scenario • Ethnography (Observation)	
2. Requirement Classification & Organization	Organize requirements into clusters Remove overlapping requirements	 Example: Hotel Management System Customer Registration Check In Room Reservation Check Out Print Report
3. Requirement Prioritization & Negotiation	Prioritize requirements Resolve conflict	
4. Requirement Documentation	Identified requirements are documented and checked for completeness, consistency, and alignment with stakeholders' expectations.	

3. Requirement Specification	(Discussed in Chapter 4) System Requirement Specification (SRS)
4. Requirement Validation	Validity: functions user wanted is correct and confirmed needed Consistency: no conflicting functions Completeness: include ALL functions needed Realism check: can implement with existing resources Verifiability: requirements are measurable? Use demo to reduce potential dispute争议 **methods can be used to validate user requirements:** prototyping/ formal technical reviews
5. Requirement Management	process that validate and meet requirements of customers/ internal/ external stakeholders Process includes(2) Management Planning: policies & procedures for requirements management are designed and specified Change Management: proposed requirements changes are analyzed and assessed

C6: Architectural Design

Architectural Design

Architectural Design Important?

- establish framework for system control and communication
- identify subsystems

Advantages of Explicit Architecture

1. Stakeholder communication- High-level presentation of system
2. System Analyst- Check whether the system can meet requirements
3. Large-scale reuse- Reusable architecture
4. Negotiation 谈判- Serve as design plan that can be used for negotiation & discussion
5. Complexity Management- Act as essential tool for complexity management

...Architectural Design Activities

1️⃣System Organization/ Structuring

- represented in block diagram/ to subsystems
- how to share data(interface, distribution, communication) between independent software unit - subsystems
- **System Organization Models (3) ...**
  - Repository model
  - Client-server model
  - Layered model

2️⃣Modular Decomposition 模组分解: Decomposition sub-system into modules

- Strategies: object-oriented/ function-oriented decomposition

3️⃣Control Modeling 控制模型...

Centralized Control: One subsystem has overall responsibility for control and starts and stops other sub-system...

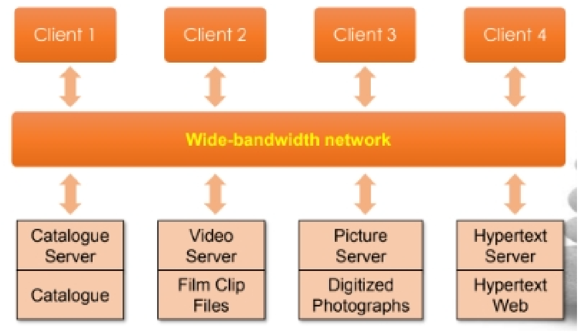
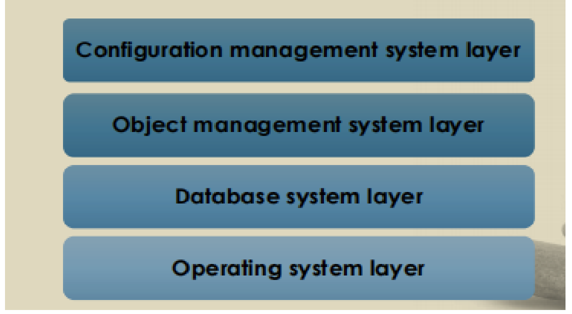
- Call-return Model
- Manager Model

Event-based Control: Response to events from other systems or environments...

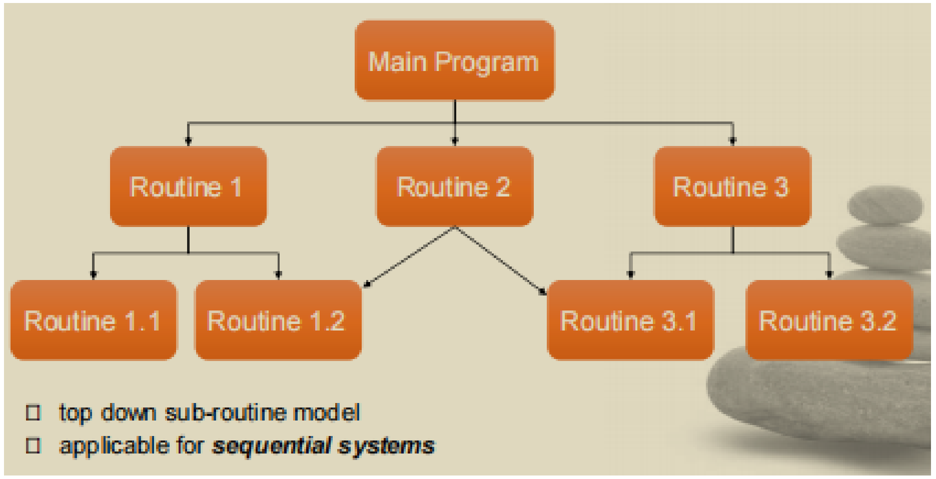
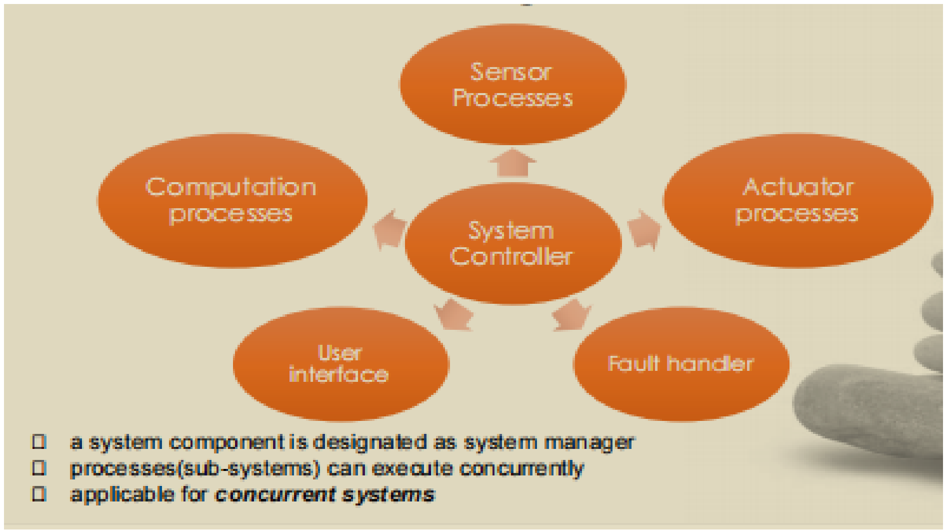
- Broadcast Model
- Interrupt-driven Model

🟦...3 System Organization Models

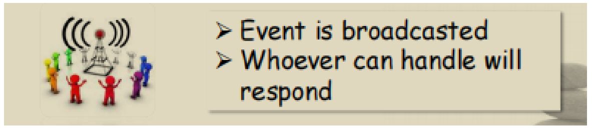
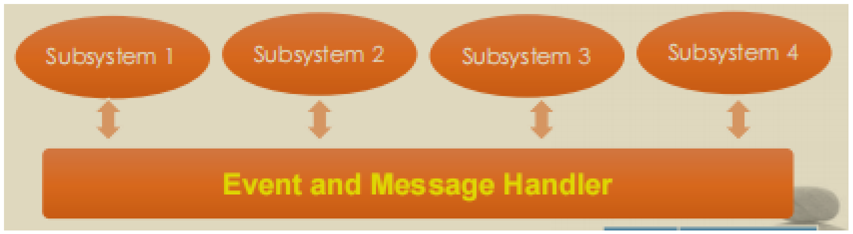
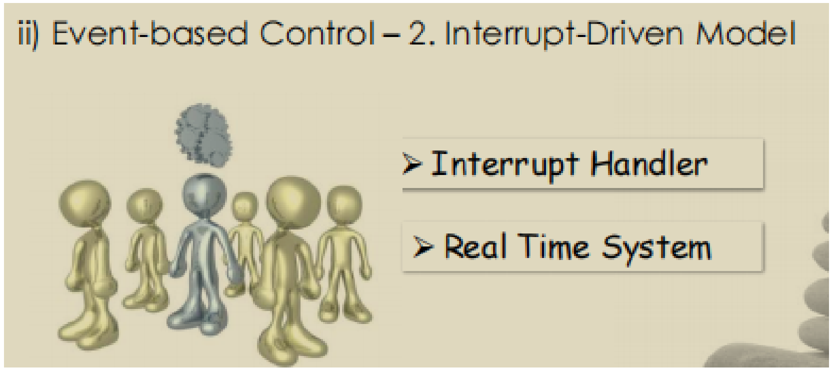
Model	Advantages ✅	Disadvantages ❌	When to use?
Repository model (central db to share data)	• Publish as repository schema 存储库模式 • Share large amount of data • Centralized management	• Difficult to distribute • Data evolution is difficult & expensive • Sub systems must compromise on a data model	• Large amount data are to be shared • Sub-system need to exchange data ○ Shared data held in central database

	• Sub systems have no concern on how data is produced 子系统不关心数据如何产生	• No scope for specific management policies	○ Sub-system maintains its own database and exchanges data by passing messages.
Client-server model (distributed system model, data and processing in different components)	• Distribution data is straightforward • Makes good use of networked systems and may only need cheaper hardware. • Easy to add or upgrade existing server	• Data exchange can be inefficient due to different data structures and no shared model. • Redundant management in each server • Hard to identify available servers and services without a central directory.	• Set stand-alone servers which provide specific services ○ Eg. printing, data management • Set of clients which call on these services • Network which allows clients to access server ii) Client-Server Model - The architecture of a film and picture library system  Client 1 Client 2 Client 3 Client 4 Wide-bandwidth network Catalogue Server Catalogue Video Server Film Clip Files Picture Server Digitized Photographs Hypertext Server Hypertext Web
Layered model (also called abstract machine model. Models the interaction between subsystems by organizing them into layers.)	• Easier to develop and maintain • Risk reduced as changes in one layer only impact neighboring layers • Issue are easier to isolate and fix within specific layers	• Slow down system performance ○ Due to additional communication between layer • Hard to implement as require significant interaction across non-neighboring layers • Limited flexibility for complex interaction	• Each layer offers specific services to the one above it. • Step-by-step development, where changes in one layer only affect the neighboring layers. iii) Layered Model – Version Management System  Configuration management system layer Object management system layer Database system layer Operating system layer

...Centralized Controll:

Call-Return Model	Manager Model
 Main Program Routine 1 Routine 2 Routine 3 Routine 1.1 Routine 1.2 Routine 3.1 Routine 3.2 □ top down sub-routine model □ applicable for ***sequential systems***	 Sensor Processes Computation processes System Controller Actuator processes User interface Fault handler □ a system component is designated as system manager □ processes(sub-systems) can execute concurrently □ applicable for ***concurrent systems***

...Event-based Control:

Broadcast Model	Interrupt-driven Model
 ➢ Event is broadcasted ➢ Whoever can handle will respond  Subsystem 1 Subsystem 2 Subsystem 3 Subsystem 4 Event and Message Handler	 ii) Event-based Control – 2. Interrupt-Driven Model ➢ Interrupt Handler ➢ Real Time System

C7: Software Testing

Fundamentals

- Testing to demonstrate presence of errors - Validation Testing: software meets requirements - Defect Testing: discover faults故障 and defects缺陷	🟦**Importance of Software Testing** - Help developers to find out the existing system errors or bugs before deploying the software into production. - Effectively check whether the developed system has fulfilled all the user and system requirements.

🟦Software Testing Stages Process (5)

Unit Testing	**Module Testing**	**Subsystem Testing**	**System Testing**	**Acceptance/ Alpha Testing**
Individual component	Collection of dependent component	Collection of modules	Integration & Compatibility 一致性 of subsystems	Final stage
● Condition ● Option	● Object Class ● Adts ● Collection of procedures/ functions		● Find discrepancies between system and (3) ○ original objectives ○ current specification ○ system documentation	● UAT/ Alpha Testing
Eg.: Student registration				
Beta Testing: Before marketed and for potential customers				

Test data	Test library
1. Live data: from DB, normal activities, difficult to get 2. Artificial data: generated to test all combinations of formats and values	● Store all test data ● Thoroughly test program

🟦Software Testing Technique

Fundamental Testing Activities (2of6)

Component Testing / Unit Testing

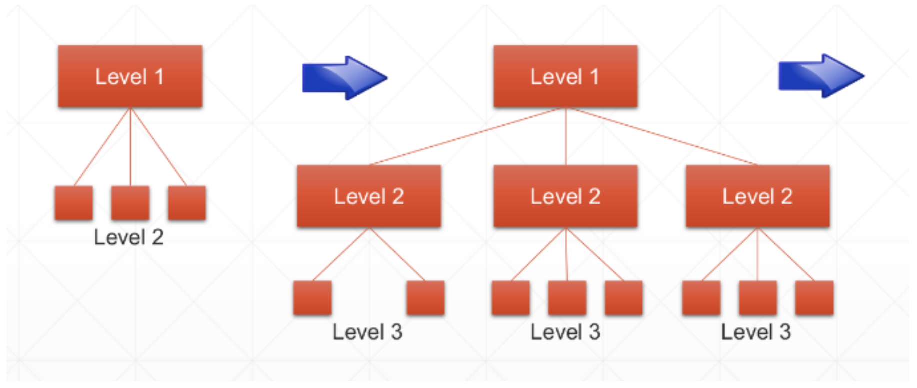
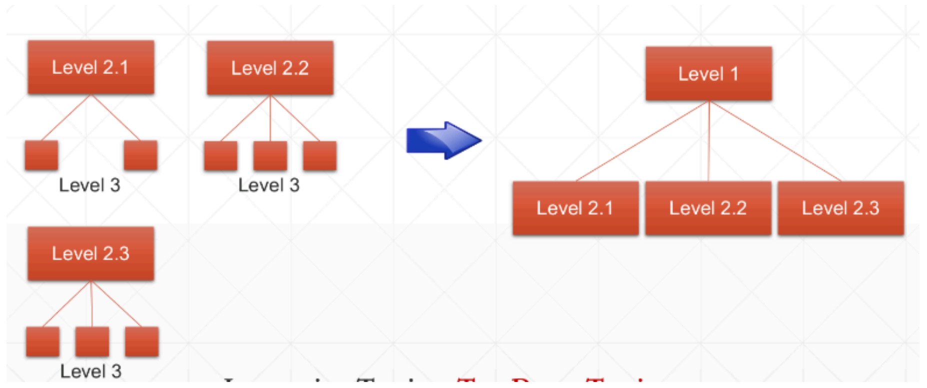
- Based on how the components are expected to work
- Testing done by the software developer.

System Testing

- Based on a written system specification
- Testing perform by independent testing team

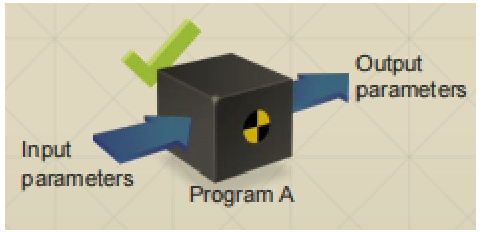
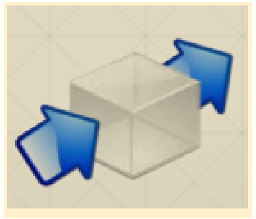
Testing Technique(4)

1️⃣Integration Testing

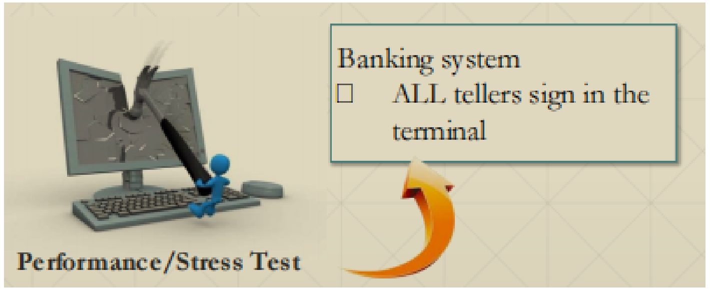
Top Down Testing	Start from upper modules and work downwards	 Level 1 Level 2 Level 1 Level 2 Level 2 Level 2 Level 3 Level 3 Level 3
Bottom Up Testing	Start from fundamental modules and work upwards	 Level 2.1 Level 3 Level 2.2 Level 3 Level 2.3 Level 3 Level 1 Level 2.1 Level 2.2 Level 2.3

Regression Testing	Rerun test for previous increments when new increment is integrated	• Will be expensive than other integration testing

2 Release Testing

Black Box Testing	• Rely on the specification of the system or component under test to derive test cases. • System behavior can only be determined by studying its inputs and associated outputs.	
White Box Testing	• Analyze code to derive test data using knowledge about component structure. • Analysis of the code can be used to find many test cases to ensure a specific level of test coverage.	

3 Performance / Stress Test

• Stressing the system by going beyond its specified limits • Test how well the system can cope with an overload situation.	 Banking system ALL tellers sign in the terminal Performance/Stress Test

4 Component / Unit Testing

i. Individual Function/Method in an Object	ii. Object Class	iii. Composite Component
• Simplest • Different input parameters • Used to design the test	• Features/attributes (setting & interrogation) • Operations • States	• Component interface interaction between parts

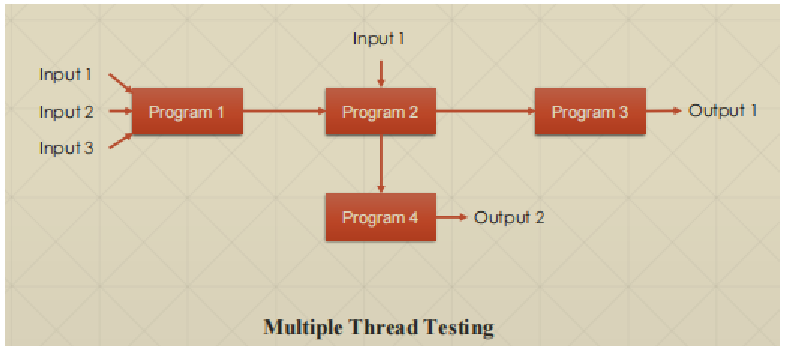
5️⃣Interface Testing

- Parameter interface
- Shared memory interface
- Procedural interface
- Message passing interface

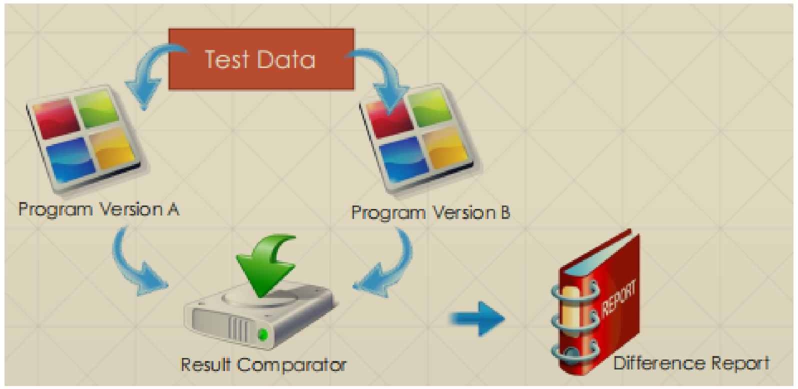
Others:

- Peak load testing
- Storage testing
- Procedure testing
- Recovery testing
- Human Factors testing
- etc

- Thread Testing



- Back-to-back testing: Version 1 vs. version 2



🟦Software Testing Strategy

A general approach to the testing process, rather than a method of work out of particular system or component tests.

Technique (2)

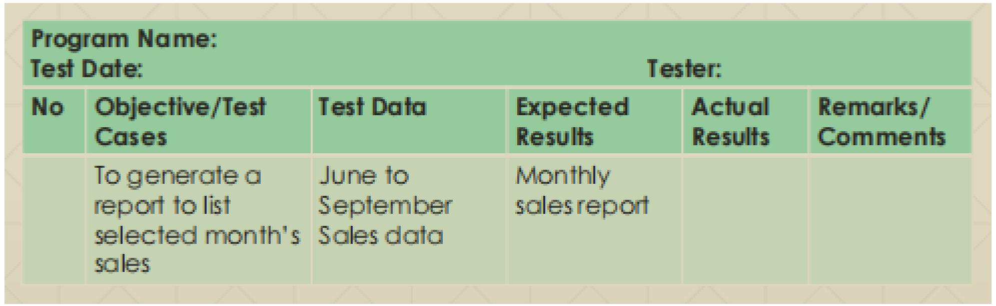
Proactive	Reactive
• Plan prevention steps before the problem happens.	• Responding to problems after they have occurred. • Eg. Fixing or managing issues after they occur

Test Case Design & Planning

Test Planning: Concerned with setting out standards for the testing process

Test Plan Contents

1. The testing process
2. Requirement traceability: ✔Unit Testing - SRS item x ✔Module Testing - SRS item y
3. Tested items: ✔Program Modules ✔User Procedures ✔Operator Procedures
4. Testing schedule: Gantt chart
5. Testing recording procedures (example)
6. Hardware & software requirements
7. Constraints: ✔test item availability ✔test resource availability ✔time constraints



Program Name:
Test Date: Tester:

No	Objective/Test Cases	Test Data	Expected Results	Actual Results	Remarks/ Comments
	To generate a report to list selected month's sales	June to September Sales data	Monthly sales report		

Approaches(3)

1. Requirements-based Testing	2. Partition Testing	3. Structural Testing
Meet system requirements	Input & Output partitions	Program statement

Testing guidelines/ principles

- All tests should be traceable to可追溯至 customer requirements
- When to start? (Test Plan: Requirement Model) (Test Cases: Design Model)
- Pareto Principles: About 80% of the results come from 20% of the causes. 大约80%的结果来自于20%的原因。
- Exhaustive详尽 testing is NOT possible

C8: User Interface Design

3 Golden Rules for a Good UI Design

1. Place user in control
2. Reduce user memory load
3. Make interface consistent

🟦UI Design Principles (6)

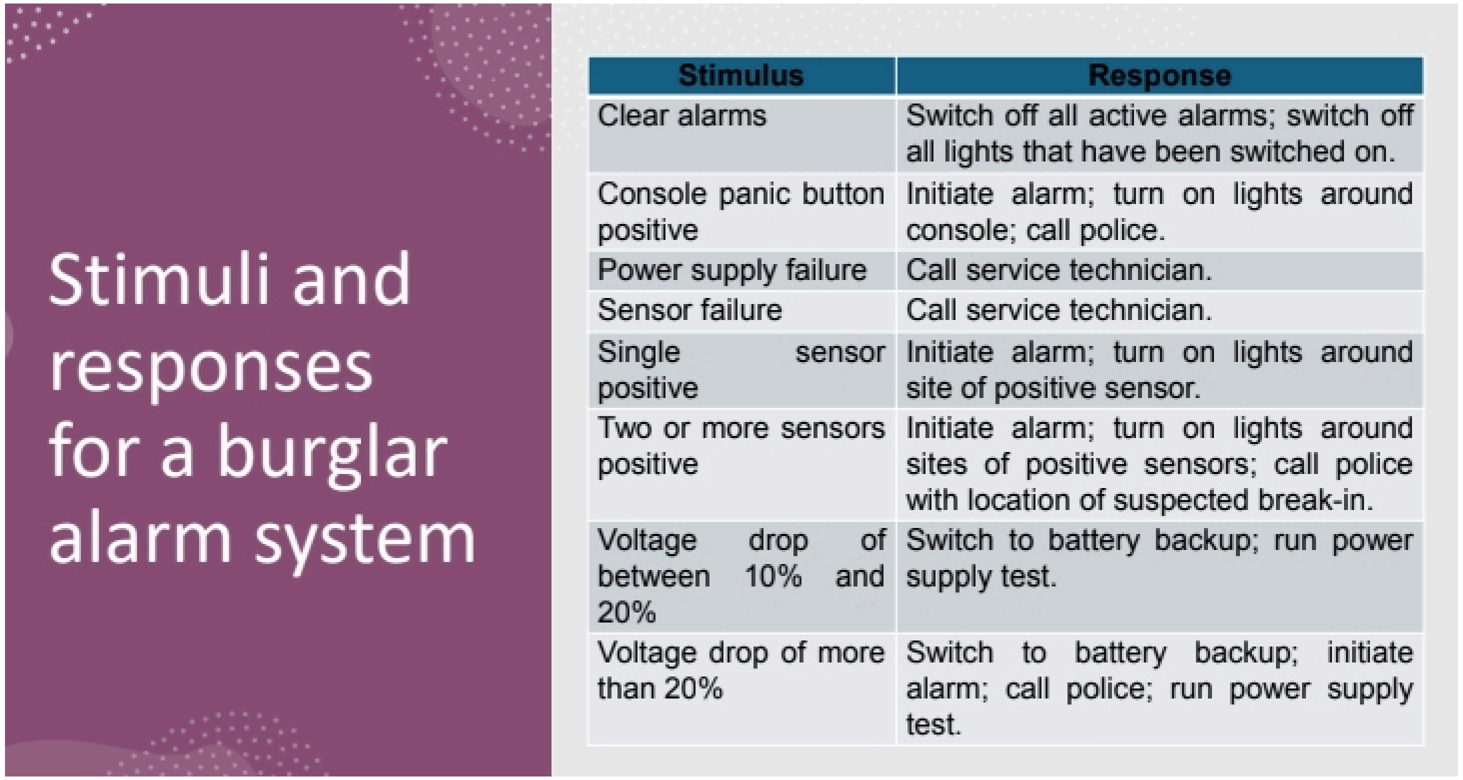
1. User Familiarity	Use terms and concepts which are drawn from the experience of the users
2. Consistency	Activate operation in the same way Eg. position of back button always located at the top right corner of the pages.
3. Minimal Surprises	User never be surprised by the system Eg. shut down immediately after done payment
4. Recoverability	Allow user to recover from errors Eg. Undo function
5. User Guidance	Provide help feature and meaningful feedback when errors occur Eg. Provide help button that introduce the function of system Eg. Prompt error message that describes what is error to the user.
6. User Diversity	Provide suitable interaction options for different types of system users. Eg. Experience user vs. Beginner Beginner: very detailed step-by-step way to perform action Eg. English Language user vs. Mandarin Language user Provide option to select preferred language

C9: REAL-TIME OPERATING SYSTEM (RTOS)

Stimulus就是一个刺激物（trigger某个response的情况）
Response就是它会给什么反馈这样咯
Two classes of Stimulus

1. Periodic Stimulus
    a. Occur at predictable time intervals (no need wait for interaction)
    b. Eg. The system will examine a sensor every 50 ms and take action depending on the sensor value
2. Aperiodic Stimulus
    a. Occur irregularly (need some interaction)

🟦Stimulus & Response



Stimuli and responses for a burglar alarm system

Stimulus	Response
Clear alarms	Switch off all active alarms; switch off all lights that have been switched on.
Console panic button positive	Initiate alarm; turn on lights around console; call police.
Power supply failure	Call service technician.
Sensor failure	Call service technician.
Single sensor positive	Initiate alarm; turn on lights around site of positive sensor.
Two or more sensors positive	Initiate alarm; turn on lights around sites of positive sensors; call police with location of suspected break-in.
Voltage drop of between 10% and 20%	Switch to battery backup; run power supply test.
Voltage drop of more than 20%	Switch to battery backup; initiate alarm; call police; run power supply test.

C10: Software Evolution

Software Evolution Process
Software Maintenance: process of modifying and updating a software system after it has been delivered to the customer.
Software Reengineering & Reverse Engineering