Software Engineering

Week 2: Software Requirements

Dr. Sridhar Iyer, IIT Bombay
Dr. Prajish Prasad, FLAME University

Software Engineering

Requirement Gathering and Analysis

Dr. Sridhar Iyer, IIT Bombay
Dr. Prajish Prasad, FLAME University

Reflection Spot

What are the disadvantages if we do not spend time and effort in identifying and documenting requirements?

Please pause the video and written down your responses

Importance of Requirement Gathering and Analysis

- Developers understand what customers want
- Customers come to an agreement about their requirements
- Increase in cost and iterations if requirements not understood properly initially

Example - Amazon Seller Portal

- Amazon wants to develop a portal for sellers
- Products which sellers list on the portal will be available for people to buy on the Seller portal

Primary Users

- Frequent users of the system

- E.g. -
  - Independent sellers
  - Sales team of consumer companies
  - Independent authors and publishers

Secondary Users

- Do not directly use the system
- Use the system through an intermediary

Tertiary Users

- Do not use the software at all
- Affected by the introduction of the software
- Influence the purchase of the software

Reflection Spot

Who are potential tertiary users of the Seller portal?

Please pause the video and written down your responses

Reflection Spot

What kind of problems or issues can arise from the gathered requirements?

Please pause the video and written down your responses

Summary

- Importance of Requirement Identification and Analysis
- Identifying requirements by considering - primary, secondary and tertiary users of the system
- Analysis of requirements essential to identify ambiguities, inconsistencies and incomplete requirements

Identifying Users and Requirements

Dr. Sridhar Iyer, IIT Bombay
Dr. Prajish Prasad, FLAME University

Recap

- Requirement Gathering and Analysis
- Users - Primary, Secondary, Tertiary

How do we gather/identify requirements?

Reflection Spot

- You have been commissioned by Amazon to build their Seller Portal

- Identified Users
  - Independent sellers
  - Sales team/manager of companies
  - Advertising department
  - Logistics/Shipping company manager
  - Buyers
  - Banks

How do we gather/identify requirements?

Requirement Gathering Techniques

- Questionnaires
- Interviews
- Focus groups and workshops
- Naturalistic observations
- Studying documentation

Questionnaires

- Series of questions designed to elicit specific information from users
- Good for getting answers to specific questions from a large group of people
- Example - Sales team managers of several mobile companies
  - What percentage of your inventory do you sell online?
  - What are the main difficulties you encounter in selling your product online?
- Used in conjunction with other techniques

Interviews

- Asking a set of questions - Face-to-face, telephonic/online interviews

- Interviews - structured, unstructured, semi-structured

Purpose of Interviews

- Getting people to explore issues
- Used early to elicit scenarios

- Example
  - What other platforms do you use to sell your products? Advantages/Disadvantages?
  - **Requirement - Tracking orders, payments, inventory, selling options**

Focus Groups

- Drawback of Interviews?
  - one perspective

- Get a group of stakeholders to discuss issues and requirements

- Advantages -
  - Gaining consensus
  - Highlighting areas of conflict, disagreement

Focus Groups - Example

- Sales team managers from different verticals - Different expectations from different industries
  - Mobile sales team - Sales summary at launch day
  - Furniture etc. - Sales summary - last 30 days

- **Requirement - Sales summary - specific day, last x days**

Naturalistic Observations

- Spending time with stakeholders as they go about their day-to-day tasks, observing their work in their natural setting

- Shadowing a stakeholder, make notes, asks questions, observe

Naturalistic Observations - Examples

- How do people sell in physical shops
  - E.g. Shopkeeper/Customer recommendations
  - E.g. Customer feedback

- Requirements -
  - **Recommendation for what other items to sell**
  - **Getting timely customer feedback**

Documentation

- Procedures and rules for a task
- Steps involved in an activity
- Regulations governing a task

- Example - Bank Regulations
  - How can you add sellers bank accounts to your portal?
  - How frequently can you deposit money to seller accounts?

Summary of requirement gathering techniques

Technique	Good for
Questionnaires	Answering specific questions
Interviews	Exploring issues
Focus groups	Collecting multiple viewpoints
Naturalistic Observations	Understanding context
Documentation	Procedures, regulations, standards

Identified Requirements

1. Add/Edit/Delete catalogue
2. Add/Edit/Delete inventory
3. Track orders
4. Track payments
5. Track inventory
6. Track sales - specific day, last x days
7. Track customer feedback

Basic requirement-gathering guidelines

- Focus on identifying stakeholders needs
- Involve all stakeholder groups
- Use combination of data gathering techniques
- Run a pilot session if possible to ensure your data-gathering session is likely to go as planned
- Data gathering is expensive, time-consuming - have to be pragmatic, make compromises

Functional and Non-Functional Requirements

Dr. Sridhar Iyer, IIT Bombay
Dr. Prajish Prasad, FLAME University

Example Requirements

- Requirement 1 - A seller can add/edit/delete their catalogue

Example Requirements

- Requirement 1 - A seller can add/edit/delete their catalogue

- Requirement 2 - When a new product is added to the catalogue, the product should appear in the catalogue within 5 sec

Reflection Spot

- Requirement 1 - A seller can add/edit/delete their catalogue
- Requirement 2 - When a new product is added to the catalogue, the product should appear in the catalogue within 5 sec

What is the difference between these two requirements?

Please pause the video and written down your responses

Requirement 1

Requirement 1 - A seller can add/edit/delete their catalogue

- Captures a functionality required by the users from the system

Requirement 1

Requirement 1 - A seller can add/edit/delete their catalogue

- Captures a functionality required by the users from the system
- f: I ➔ O

Non-functional Requirements

- Non-functional requirements essentially specifies how the system should behave

Reflection Spot

What are some non-functional requirements for the Amazon Seller Portal?

Please pause the video and written down your responses

Reliability

- **Reliability** is the extent to which a program behaves the same way over time in the same operating environment

Robustness

- **Robustness** is the extent to which a program can recover from errors or unexpected input

Summary

- Functional requirements describe what the system should do

- Non-functional requirements essentially specifies how the system should behave
  - E.g. Reliability and Robustness, Performance, Portability, Security

Software Requirement Specification

Dr. Sridhar Iyer, IIT Bombay
Dr. Prajish Prasad, FLAME University

Recap

- Requirement gathering
- Functional and non-functional requirements
- This and upcoming lectures - organize these requirements

Plan and Document perspective

- Requirement gathering and analysis - done by system analyst, along with other members of the software team

- Organize these requirements - **Software Requirements Specification (SRS)** document

SRS Document

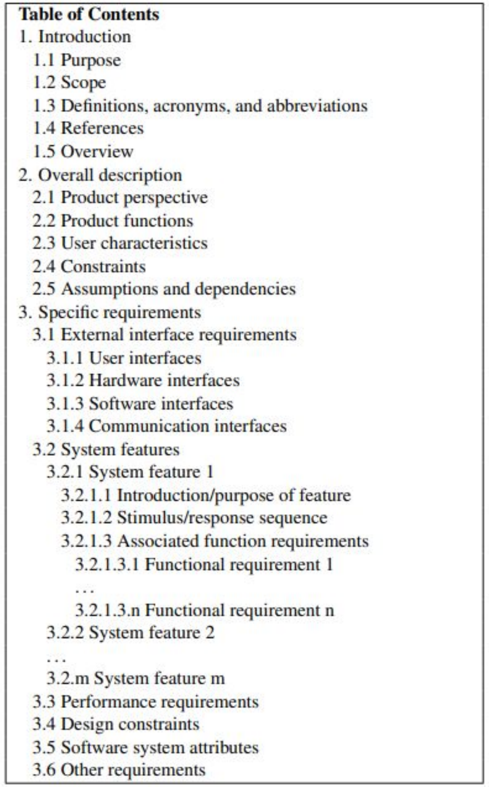


Table of Contents

1. Introduction
   - 1.1 Purpose
   - 1.2 Scope
   - 1.3 Definitions, acronyms, and abbreviations
   - 1.4 References
   - 1.5 Overview
2. Overall description
   - 2.1 Product perspective
   - 2.2 Product functions
   - 2.3 User characteristics
   - 2.4 Constraints
   - 2.5 Assumptions and dependencies
3. Specific requirements
   - 3.1 External interface requirements
     - 3.1.1 User interfaces
     - 3.1.2 Hardware interfaces
     - 3.1.3 Software interfaces
     - 3.1.4 Communication interfaces
   - 3.2 System features
     - 3.2.1 System feature 1
       - 3.2.1.1 Introduction/purpose of feature
       - 3.2.1.2 Stimulus/response sequence
       - 3.2.1.3 Associated function requirements
         - 3.2.1.3.1 Functional requirement 1
         - . . .
         - 3.2.1.3.n Functional requirement n
     - 3.2.2 System feature 2
     - . . .
     - 3.2.m System feature m
   - 3.3 Performance requirements
   - 3.4 Design constraints
   - 3.5 Software system attributes
   - 3.6 Other requirements

SRS Document

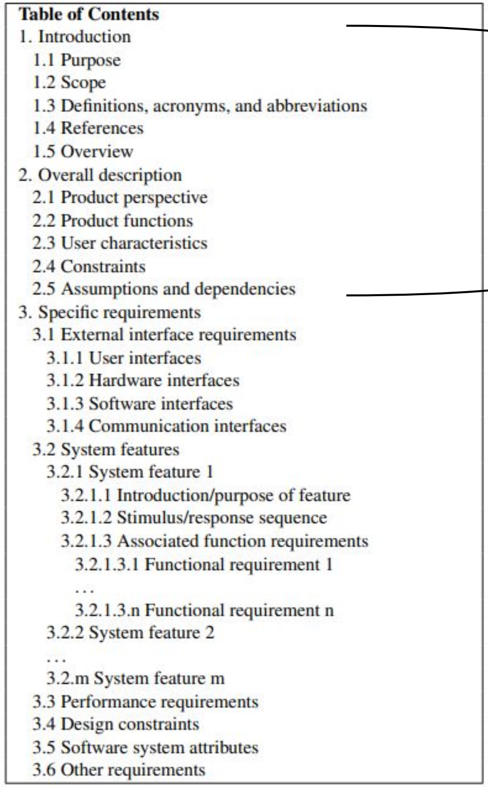


Table of Contents

1. Introduction
   - 1.1 Purpose
   - 1.2 Scope
   - 1.3 Definitions, acronyms, and abbreviations
   - 1.4 References
   - 1.5 Overview
2. Overall description
   - 2.1 Product perspective
   - 2.2 Product functions
   - 2.3 User characteristics
   - 2.4 Constraints
   - 2.5 Assumptions and dependencies
3. Specific requirements
   - 3.1 External interface requirements
     - 3.1.1 User interfaces
     - 3.1.2 Hardware interfaces
     - 3.1.3 Software interfaces
     - 3.1.4 Communication interfaces
   - 3.2 System features
     - 3.2.1 System feature 1
       - 3.2.1.1 Introduction/purpose of feature
       - 3.2.1.2 Stimulus/response sequence
       - 3.2.1.3 Associated function requirements
         - 3.2.1.3.1 Functional requirement 1
         - . . .
         - 3.2.1.3.n Functional requirement n
     - 3.2.2 System feature 2
     - . . .
     - 3.2.m System feature m
   - 3.3 Performance requirements
   - 3.4 Design constraints
   - 3.5 Software system attributes
   - 3.6 Other requirements

Broad outline and description of the software system

SRS Document

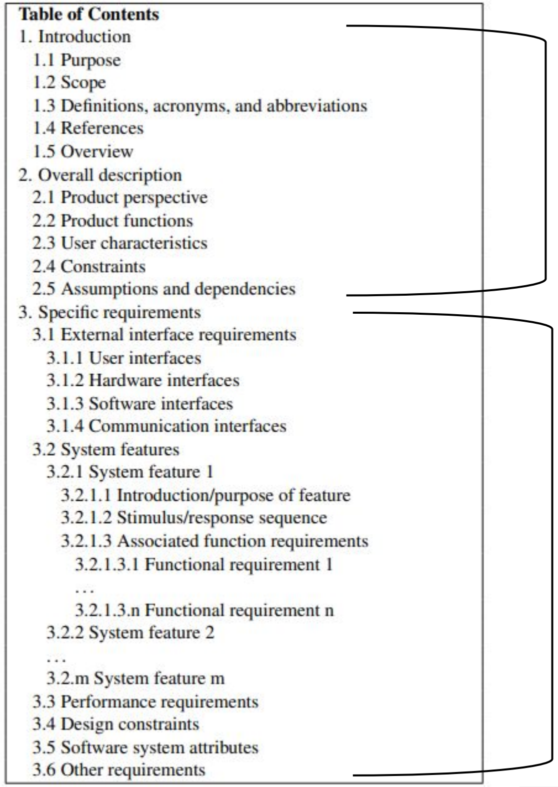


Table of Contents

1. Introduction
   - 1.1 Purpose
   - 1.2 Scope
   - 1.3 Definitions, acronyms, and abbreviations
   - 1.4 References
   - 1.5 Overview
2. Overall description
   - 2.1 Product perspective
   - 2.2 Product functions
   - 2.3 User characteristics
   - 2.4 Constraints
   - 2.5 Assumptions and dependencies
3. Specific requirements
   - 3.1 External interface requirements
     - 3.1.1 User interfaces
     - 3.1.2 Hardware interfaces
     - 3.1.3 Software interfaces
     - 3.1.4 Communication interfaces
   - 3.2 System features
     - 3.2.1 System feature 1
       - 3.2.1.1 Introduction/purpose of feature
       - 3.2.1.2 Stimulus/response sequence
       - 3.2.1.3 Associated function requirements
         - 3.2.1.3.1 Functional requirement 1
         - . . .
         - 3.2.1.3.n Functional requirement n
     - 3.2.2 System feature 2
     - . . .
     - 3.2.m System feature m
   - 3.3 Performance requirements
   - 3.4 Design constraints
   - 3.5 Software system attributes
   - 3.6 Other requirements

Broad outline and description of the software system

Functional and non-functional requirements

SRS Document

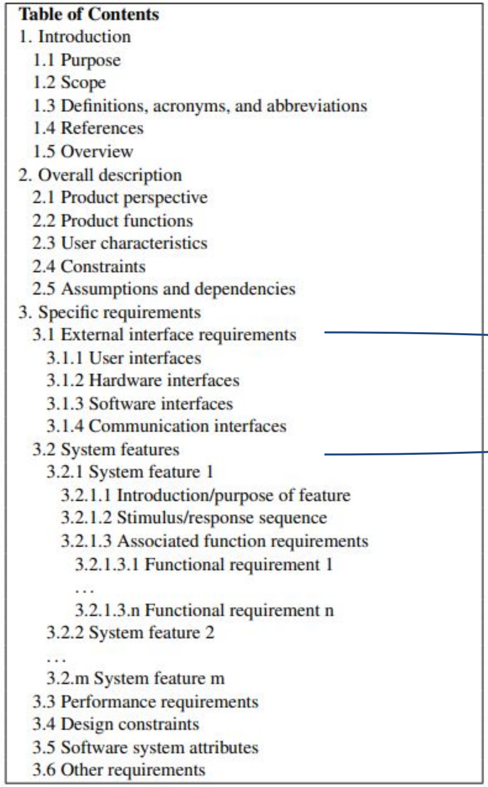


Table of Contents

External Interface requirements -

- User interfaces (UI)
- Hardware interface
- Software interface - connection between other software components
- Communication interfaces

SRS Document

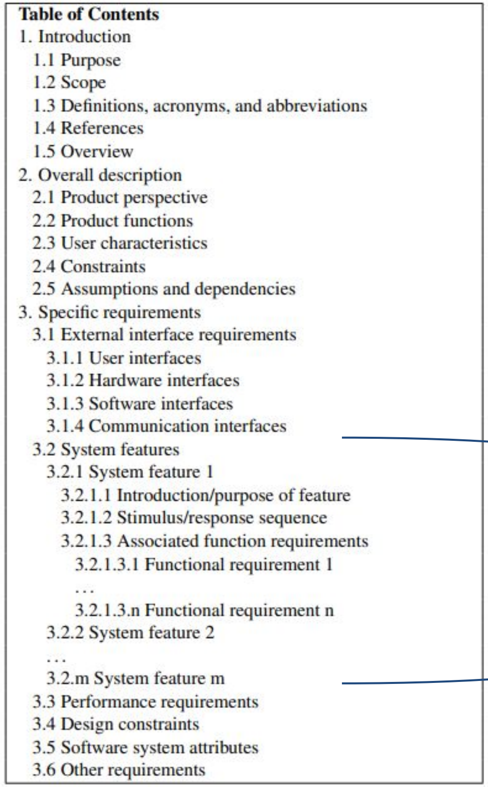


Table of Contents

Example of system features:

1. Manage catalogue
2. Manage inventory
3. Track orders
4. Track payments
5. Track inventory
6. Track sales - specific day, last x days
7. Track customer feedback

SRS Document

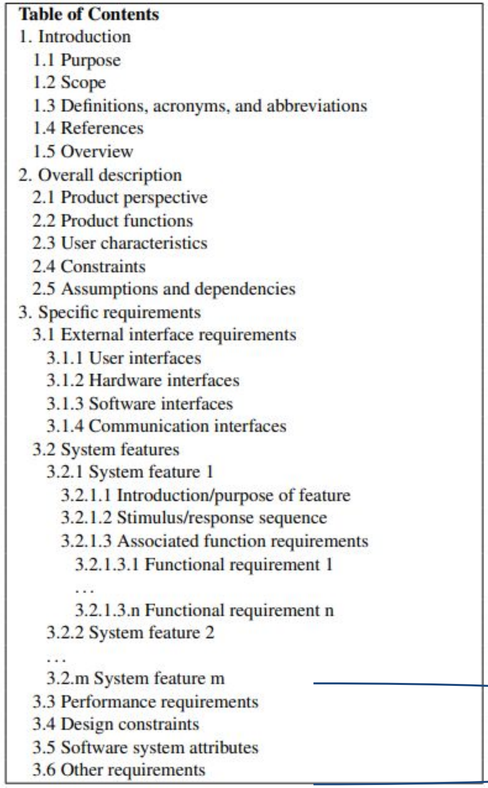


Table of Contents

Reflection Spot

What are the advantages of maintaining an SRS document?

Please pause the video and written down your responses

SRS Advantages

- Forms an agreement between customers and developers
- Reduces future reworks
- Provides a basis for estimating costs and schedules
- Facilitates future extensions

Summary

- Software Requirement Specification (SRS) document
- Drawback - Lot of documentation!! Good if the requirements are fixed
- Agile perspective - Behaviour driven design - address this drawback

Software Engineering

Behaviour Driven Design - User Stories

Dr. Sridhar Iyer, IIT Bombay
Dr. Prajish Prasad, FLAME University

Recap

- Requirement phase - Plan and Document perspective
    - SRS Document

Customers unsure of the requirements??

$\Downarrow$

Can be addressed by the Agile Perspective

Behaviour Driven Design (BDD)

- Asks questions about the behaviour of an application **before and during development**
- Requirements are continuously refined to meet user expectations
- BDD Version of requirements - **User Stories**

User Stories

- Short, informal, plain language description of what a user wants to do within a software product which is of value for them

- Smallest unit of work which can be done in 1 sprint

- Role-feature-benefit pattern/template
  - As a **[type of user]**,
  - I want **[an action]**,
  - So that **[a benefit/value]**

User Stories Examples

View inventory

```
Feature: View inventory

As an independent seller,

I want to view my inventory

So that I can take stock of
products which are low in number
```

User Stories Examples

Track customer feedback

Feature: Track customer feedback

As an **independent seller,**

I want to **view my customers' feedback for each product**

So that **I can get a sense of pertinent issues in my products**

Benefits of User Stories

- Lightweight
- Help plan and prioritize development

```
Feature: View inventory
```

```
Feature: Track customer feedback
```

Developers to customers - will provide “View inventory feature in 2 weeks”

Benefits of User Stories

- Concentrate on behaviour vs implementation of the application
- Conversation between users and the development team

```
Feature: Track customer feedback

As an independent seller,

I want to view my customers' feedback for
each product

So that I can get a sense of pertinent
issues in my products
```

"I would also like to be notified by email about 1 or 2 star ratings of my product"

Characteristics of User Stories - SMART

- Specific
- Measurable
- Achievable
- Relevant
- Timeboxed

Specific User Stories

- Specific - know exactly what to implement

Vague

```
Feature: User can search
for a product in the
catalogue
```

Specific

```
Feature: User can search
for a product by title in
the catalogue
```

Measurable User Stories

- Known expected results for some inputs

Not measurable

```
Feature: The Seller
portal should have good
response time
```

Measurable

```
Feature: When adding a
product to the catalogue,
the product should appear
in the catalogue within 3
seconds
```

Achievable User Stories

- Ideally - Implement the user story in one agile iteration (1-2 weeks)
- If not possible - subdivide stories into smaller ones

```
Feature 1: View
inventory
```

```
Feature 1a: View all products
in inventory in a single page
```

```
Feature 1b: Add pagination and
filters
```

Relevant User Stories

- Relevant - Business value to one or more stakeholders

- Ask questions
  - “Why”
  - “So what”

Timeboxed User Stories

- Stop implementing a feature once time budget exceeded

- Options?
  - Give up
  - Divide the story into smaller ones
  - Reschedule what is left

Reflection Spot

What are drawbacks of User Stories?

Please pause the video and written down your responses

Drawbacks - User Stories

- Continuous contact with users not possible
- Not scale to very large projects, safety critical applications

Summary

- Behaviour driven development (BDD)

- BDD version of requirements - User stories
  - As a **[type of user]**,
  - I want **[an action]**,
  - So that **[a benefit/value]**

- SMART User stories

- Benefits, Drawbacks

Week 1: Deconstructing the Software Development Process

Dr. Sridhar Iyer, IIT Bombay
Dr. Prajish Prasad, FLAME University

Software Engineering

Thinking of Software in terms of Components

Dr. Sridhar Iyer, IIT Bombay
Dr. Prajish Prasad, FLAME University

Reflection Spot

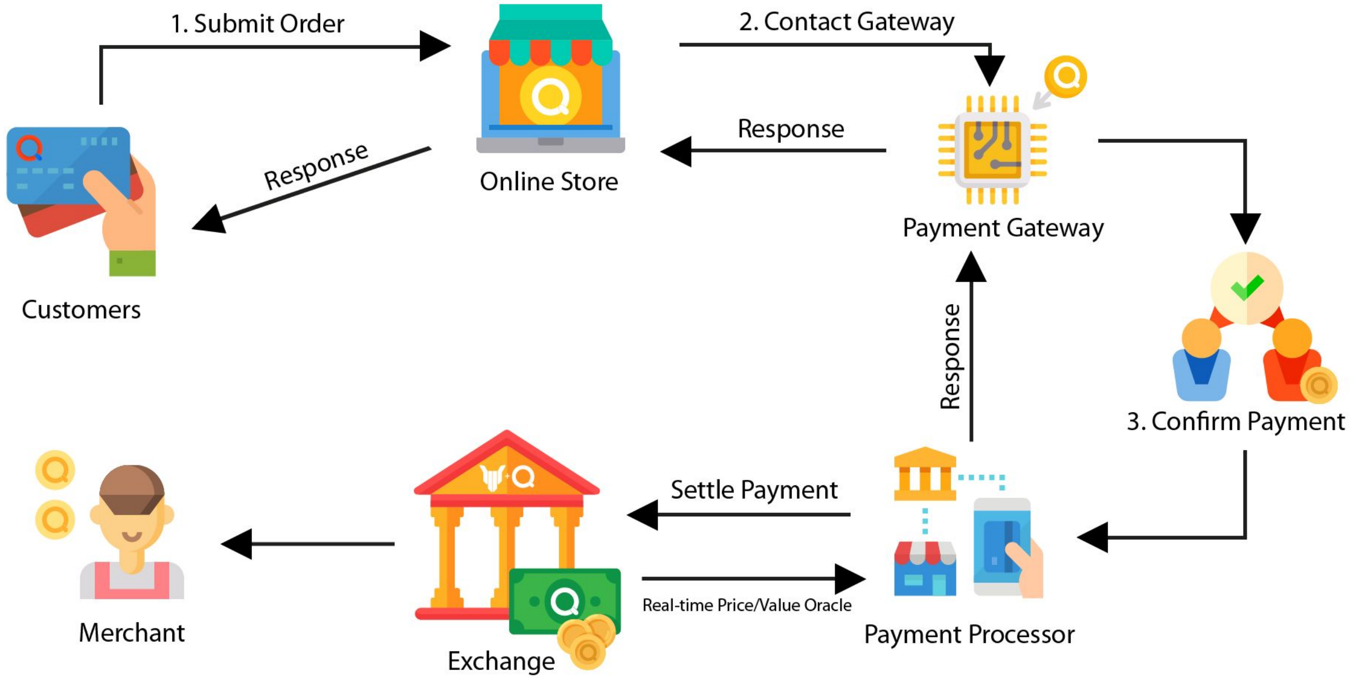
We discussed that Amazon has several components. Could you list a few of them ?

Please pause the video and write down your responses

Inventory Management

- Inventory gets updated based on current purchasing and seasonal trends
- My homepage is customised based on my shopping and viewing history

Payment Gateway



https://cdn.truelancer.com/upload-original/1658216-Payment-Gateway-Process.png

Summary

- Discussed various components of a large software system E.g. Amazon
    - Inventory Management System
    - Payment Gateway

- Software can be divided into separately addressable components called **modules** that are integrated to satisfy requirements

Software Engineering

Software Development Process - Requirement Specification

Dr. Sridhar Iyer, IIT Bombay
Dr. Prajish Prasad, FLAME University

Reflection Spot

What do you think is the first step in creating a new software component?

Please pause the video and write down your responses

First Step in Creating Software

First Step in Creating Software

- Study existing components of the system -
  to understand how the new component will interact with existing components

First Step in Creating Software

- Look at similar systems to understand features e.g. PayTM, PhonePe

Requirement Specification

Requirements

- Goals the implemented system should have
- Should cater to the need of clients

Client - External User

Example - Mobile banking software serves bank customers

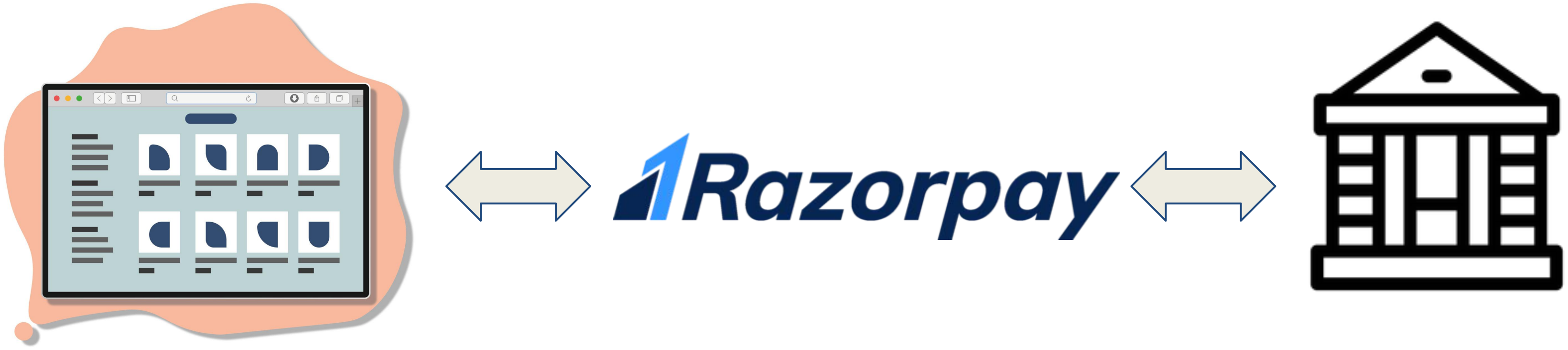
Client - Internal to your Company

Example -
Building an internal employee resource portal

- Internal products team → To build this portal

Client - Another Software

Example -
A payment gateway interfaces with another ecommerce system



Clients

- Think about **who** is going to use your software, for **what purpose**, and in **what way**

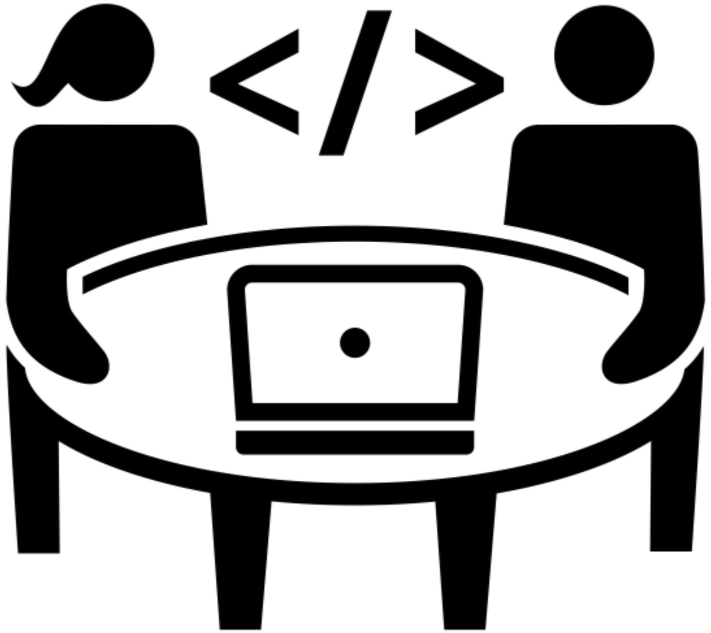
Summary

- **Requirement specification** - First step in the software development process
- **Clients** - end users of the software
- Need to ensure that the requirements capture clients' needs

Software Design and Development

Dr. Sridhar Iyer, IIT Bombay
Dr. Prajish Prasad, FLAME University

Software Development Team



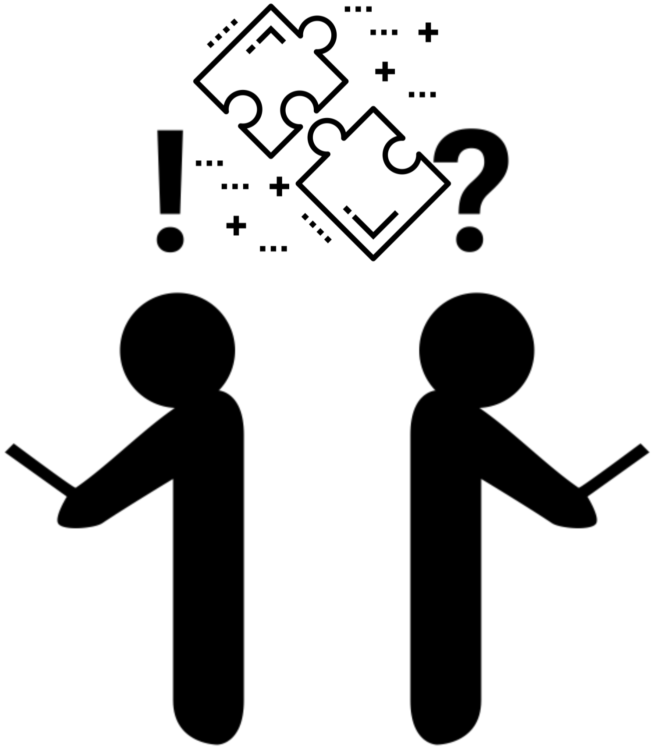
Reflection Spot

What difficulties are you likely to encounter if you directly start coding?

Please pause the video and write down your responses

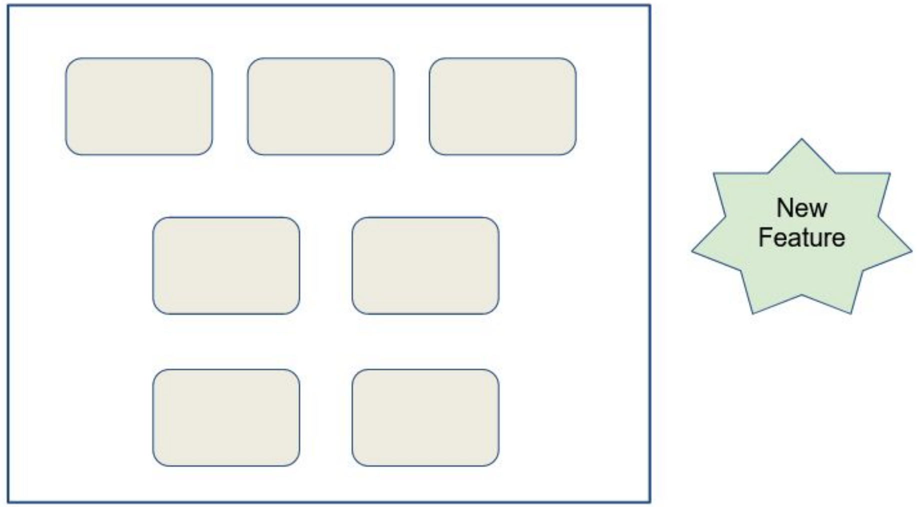
Issues during integration

- Different developers may have different ideas about how the functionality should be implemented



Difficulties while adding new features

- Adding new features - big picture view of the system is necessary



Software Design

Requirements

- Goals the implemented system should have
- Should cater to the need of clients

Design

- Big picture view of the software system
- Provides a structure to the software system

Reflection Spot

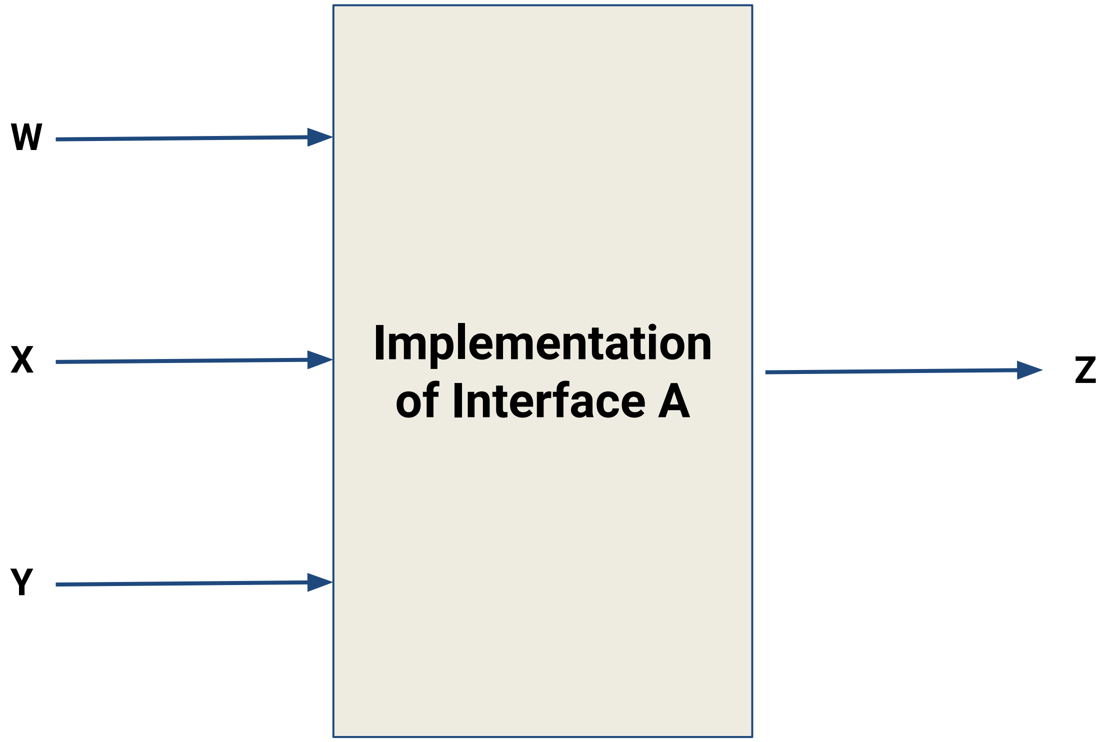
How do you think people work in the development phase?

Please pause the video and write down your responses

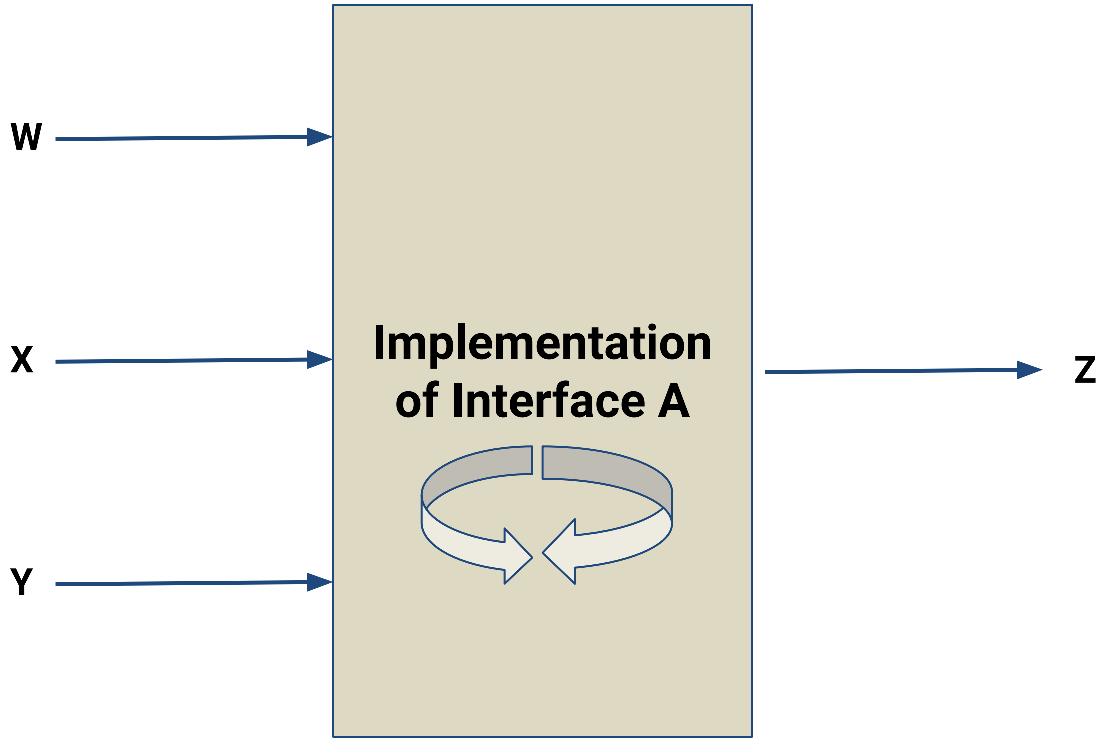
Development Phase

Development Phase

Development Phase



Development Phase



Software Development

Requirements

- Goals the implemented system should have
- Should cater to the need of clients

Design

- Big picture view of the software system
- Provides a structure to the software system

Development

- Write code based on the requirements and the design
- Usually distributed
- Developer documentation and precise interface definitions

Software Engineering

Software Testing and Maintenance

Dr. Sridhar Iyer, IIT Bombay
Dr. Prajish Prasad, FLAME University

Reflection Spot

Why do you think testing is necessary? What can go wrong if we release the software directly?

Please pause the video and write down your responses

Importance of Testing

- Testing is done to ensure that the software behaves according to the requirements

- Many bugs might still exist in the system

Importance of Testing

- In 2002 - $59.5 billion in losses [1]
  In 2016 - $1.1 trillion [2]

- A failure to address bugs can even cause severe catastrophes

[1] Newman, M., 2002. Software errors cost us economy 59.5 billion annually. NIST Assesses Technical Needs of Industry to Improve Software-Testing
[2] https://medium.com/@ryancohane/financial-cost-of-software-bugs-51b4d193f107

Testing

- Unit Testing
- Integration Testing
- Acceptance Testing

Testing Methodologies

Alpha Testing

Conducted by: internal employees in a lab/staging environment

Goal: catch as many issues as possible before the product has been released to the public

Testing Methodologies

Alpha Testing

Conducted by: internal employees in a lab/staging environment

Goal: catch as many issues as possible before the product has been released to the public

Beta Testing

Conducted by actual users in a real-live setting

Maintenance Phase

Maintenance - After the feature is rolled out, monitor how users are using the feature

Purposes of maintenance

- Monitor what users are doing, and how they are using the software.
- Change the code for upgrades/updates
- Add features

Software Development Process

Requirements

- Goals the implemented system should have
- Should cater to the need of clients

Design

- Big picture view of the software system
- Provides a structure to the software system

Development

- Write code based on the requirements and the design
- Usually distributed

Testing

- Ensures that the software behaves according to the requirements

Maintenance

- Monitor what users are doing
- Change code for updates

Software Development Models - Plan and Document Perspectives

Dr. Sridhar Iyer, IIT Bombay
Dr. Prajish Prasad, FLAME University

Beginning of Software Engineering Discipline

- Experience in previous projects
  - "Build and Fix"

- Good Principles and practices + Research Innovations ➔ Software Engineering

- Well-defined set of activities -
  - Software life cycle model
  - Software Development Life cycle (SDLC)
  - Software development process model

Software Lifecycle

Different stages/phases/activities over which a software evolves from the initial customer request to a fully developed software

Software Development Lifecycle

Requirements

- Goals the implemented system should have
- Should cater to the need of clients

Design

- Big picture view of the software system
- Provides a structure to the software syste

Development

- Write code based on the requirement
- Usually distributed

Testing

- Ensures that the software behaves according to the requirements

Maintenance

- Monitor what users are doing
- Change code for updates

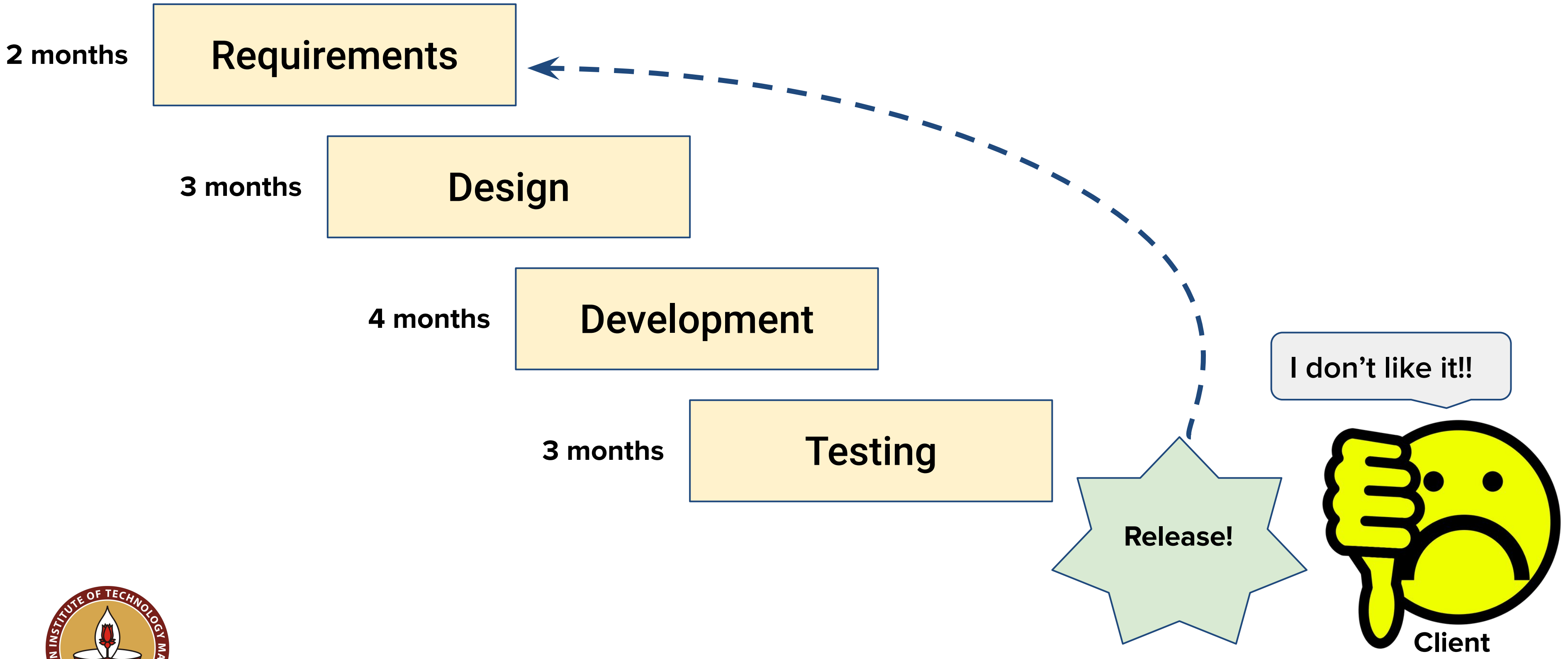
Plan and Document Perspective - Waterfall model

Reflection Spot

What could go wrong if we follow these phases sequentially?

Please pause the video and written down your responses

Drawbacks of the Waterfall Model



Drawbacks of the Waterfall Model

- Increase in cost, time if changes are required later on
- Clients may not know what they need!
- Designers may not know which design might be the most feasible/usable by clients
- Quite long - usually takes 6-18 months for 1 cycle

Reflection Spot

How can we address this issue of the waterfall model?

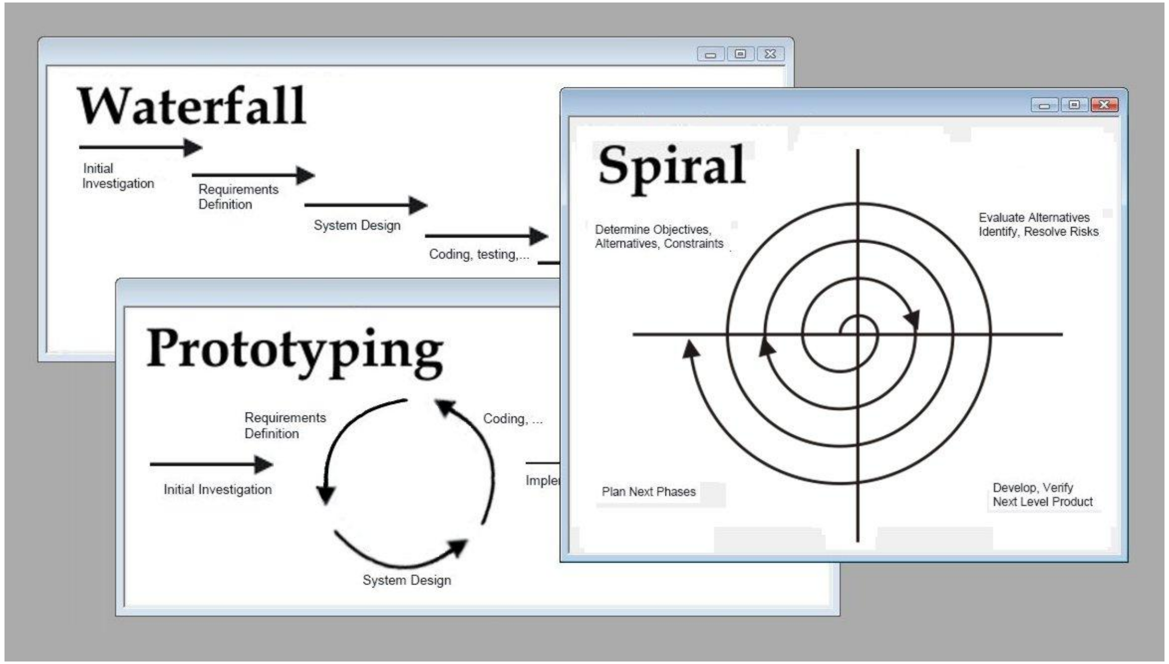
Please pause the video and written down your responses

Prototype Model

- Build a working prototype before development of the actual software
- Prototype - usually not used later
- Advantages -
  - Exact form of solution and technical issues are unclear
  - Useful to get feedback from customers
- Disadvantages -
  - Increased development costs
  - Bugs can appear later in the development cycle

Spiral Model

- Incrementally build the software and get feedback, refine



Summary

- Software development lifecycle

- Different models in the plan and document perspective -
    - Waterfall
    - Prototype
    - Spiral

Software Engineering

Software Development Models - Agile Perspective

Dr. Sridhar Iyer, IIT Bombay
Dr. Prajish Prasad, FLAME University

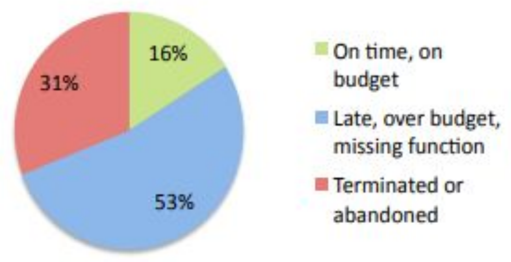
Recap

- Software development lifecycle

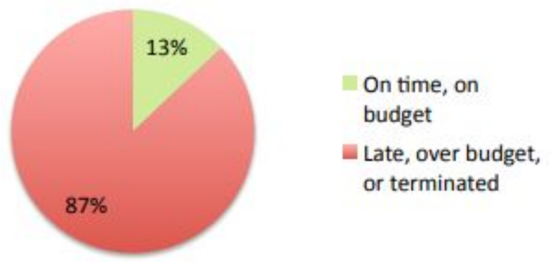
- Different models in the plan and document perspective -
  - Waterfall
  - Prototype
  - Spiral

Study of Software Projects

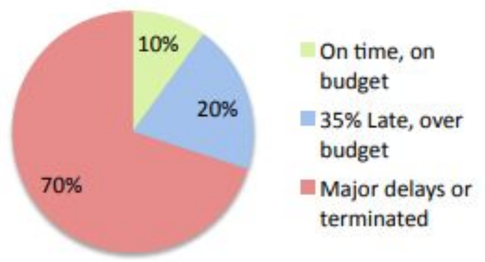
a) Software Projects (Johnson 1995)



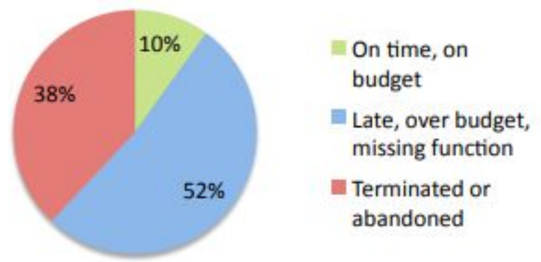
b) Software Projects (Taylor 2000)



c) Software Projects (Jones 2004)



d) Software Projects (Johnson 2013)

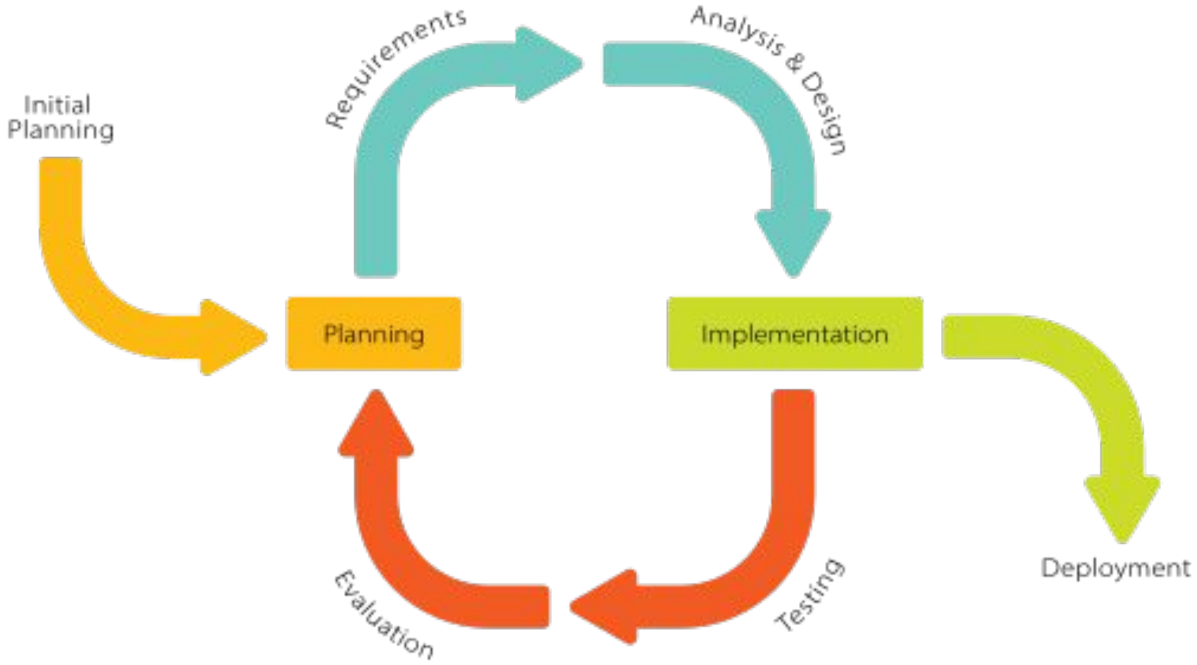


Engineering Software as a Service: An Agile Approach Using Cloud Computing (saasbook.info) - Ch1 - pg 12

Agile Manifesto

- 4 key principles -
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan

Incremental Development



Agile Approaches

- Extreme Programming (XP)
- Scrum
- Kanban

Agile Practices

- User stories
- Sprints
- Scrum Stand-up meetings
- Test-driven development

Agile Philosophy

- Rather than just following approaches and practices - more of adhering to the broad philosophy
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan

Reflection Spot

Plan and Document vs Agile - When to use?

Please pause the video and written down your responses

When to use Agile/Plan and Document

	Question: A no answer suggests Agile; a yes suggests Plan and Document
1	Is specification required?
2	Are customers unavailable?
3	Is the system to be built large?
4	Is the system to be built complex (e.g., real time)?
5	Will it have a long product lifetime?
6	Are you using poor software tools?
7	Is the project team geographically distributed?
8	Is team part of a documentation-oriented culture?
9	Does the team have poor programming skills?
10	Is the system to be built subject to regulation?

I. Sommerville. Software Engineering, Ninth Edition. Addison-Wesley, 2010. ISBN 0137035152.