Software Engineering

Bsc. CSIT and BCA

Compiled by : T.R. JOSHI

Syllabus

- Unit -1: Introduction
- Unit - 2: Software processes
- Unit - 3: Agile software development
- Unit - 4: Requirement engineering
- Unit - 5: System modeling
- Unit - 6: Architectural Design
- Unit - 7: Design and implementation
- Unit - 8: Software Testing
- Unit - 9: Software Evolution
- Unit - 10: Software management

Unit : 2 - Software Processes: Contents

- Software Process
- Software Process Models
- Waterfall Model
- Incremental Development
- Integration and Configuration
- Software Process Activities
- Software Specification
- Software Design and Implementation
- Software Validation
- Software Evolution
- Coping with Change (Prototyping, Incremental Delivery);
- Process Improvement

Introduction

- A software process is a set of related activities that leads to the production of a software system.
- The process used in different companies depends on the type of software being developed, the requirements of the software customer, and the skills of the people writing the software.
- Four fundamental processes are:
  - Software specification
  - Software development
  - Software deployment
  - Software evolution

The Software Process

- A structured set of activities required to develop a software system
- Software process involves:
  - Specification: what system should do?
  - Design and implementation: organization of system
  - Validation : It does what customer wants
  - Evolution: Changing the system in response
- Process description includes:
  - Products: outcome of process activity after execution
  - Roles: roles of people involved in process

Software Specification:

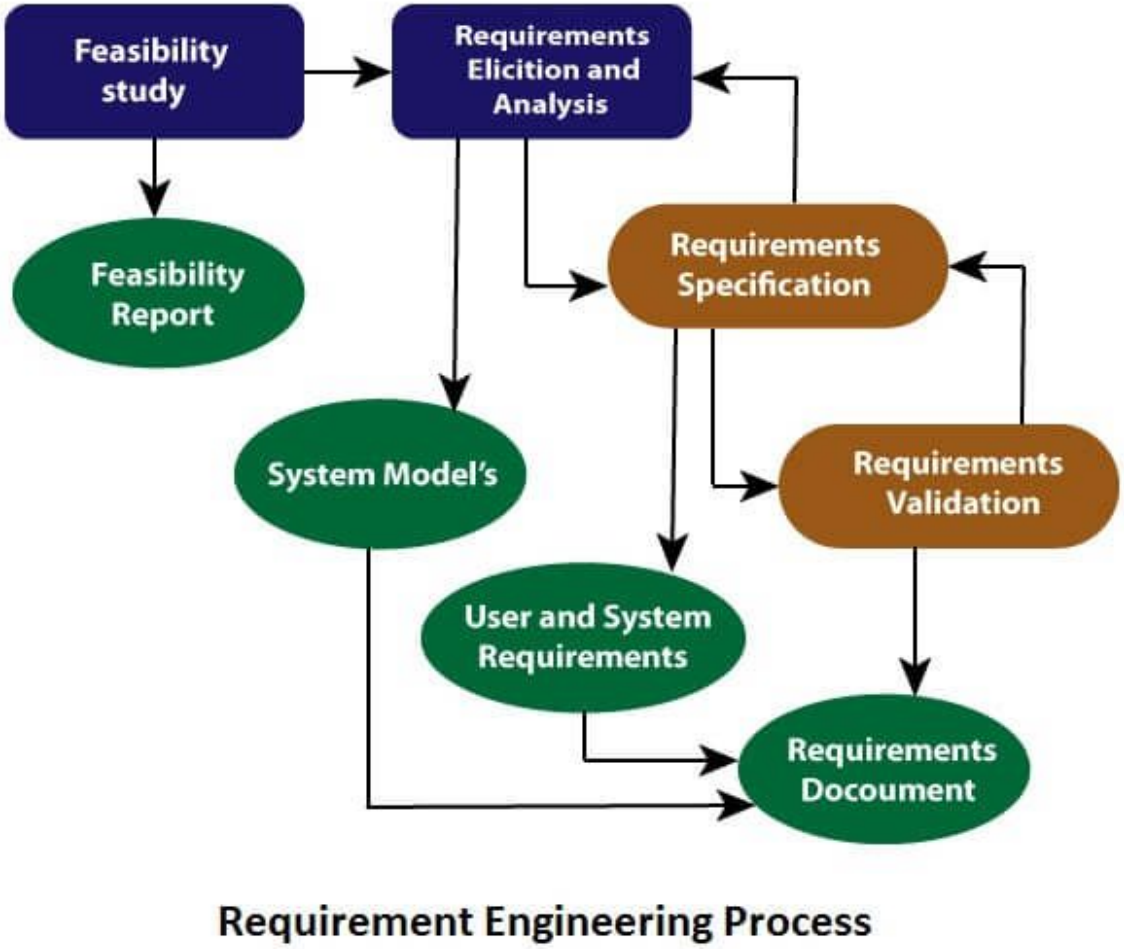
- Specifies what services are required from the system
- Specification or requirement involves:
- Feasibility study
  - Technically and economically feasible or not?
  - ROI, market share, urgency of development?
- Requirements elicitation and analysis
  - What stakeholders expect and require?
  - Client is satisfied or not?

Software Specification:

- **Requirements specification:**
  - **Defining the requirement in detail**
- **Requirement Validation:**
  - **Are they feasible, testable, sufficient , necessary ..?**

Requirement

Specification



Requirement Engineering Process

Design and Implementation:

- Converting system specification into an executable system
- **Software Design:**
  - Design the structure/Architecture
- **Implementation:**
  - Translate the model into an executable app
  - Programming refers to the implementation of design

Design Activities:

- **Architectural Design:**
  - Represents overall structure of the system
  - Physical Components: subsystem or module and the relationship
- **Interface Design :**
  - Interface between the system components
  - Recognize to the user that UI is the application
- **Component Design:**
  - Each subpart or the components of the system
- **Database Design:**
  - designs the system data structure

Software Validation

- **Verification and Validation:**
  - **Conforms that system meets its specification and the requirements of customers**
- **Verification is a process of determining if the software is designed and developed as per the specified requirements.**
- **Validation is the process of checking if the software (end product) has met the client's true needs and expectations.**

Stages Of Testing:

- **Component Testing:**
- **Individual Component are tested**
- **Component may be functions or objects**
- **System Testing**
- **Testing whole system**
- **Acceptance Testing**
- **Testing with customer to check**

Software Process Models

- **Software process model , sometimes can also be referred as SDLC**
- **Every model can have particular approach of providing information about the process, but individual model is not perfect for all kinds of applications.**
- **Some of the general model are:**
  - **The waterfall model**
  - **Incremental model**
  - **Evolutionary Model**

A. Waterfall Model

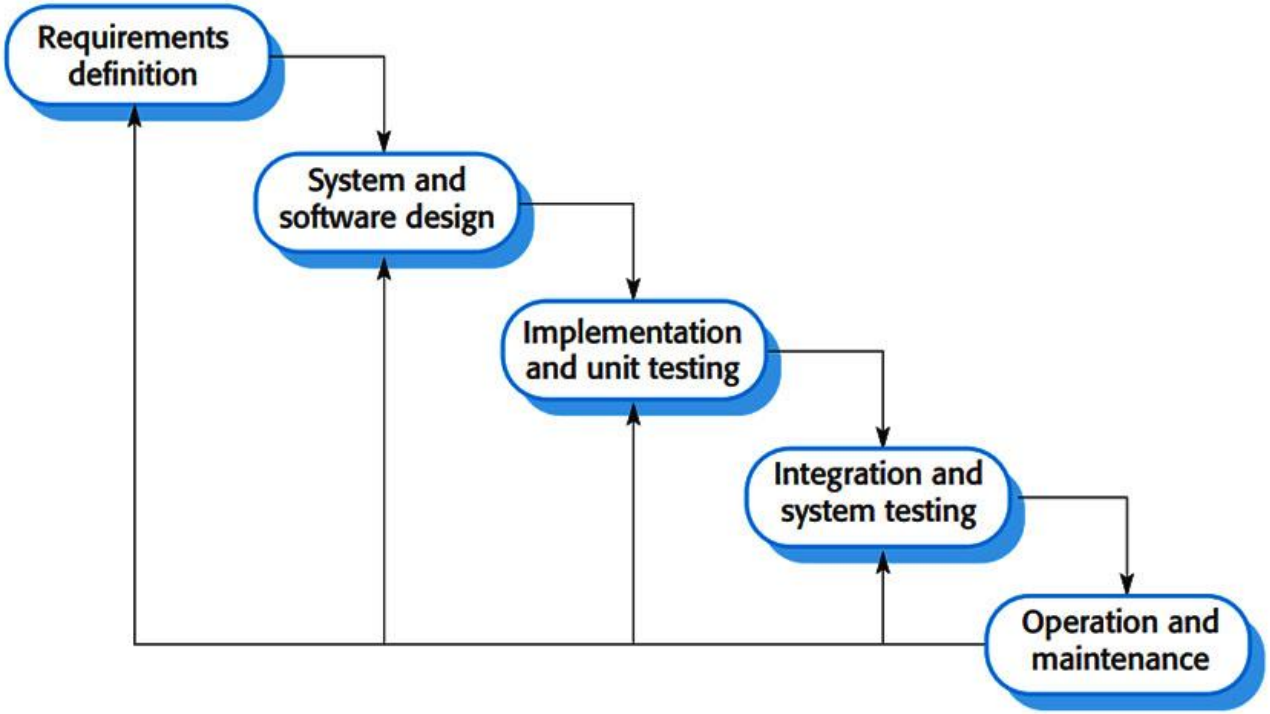


Figure 2.1 The waterfall model

A. Waterfall Model

- Requirement analysis and definition : System goals and requirement specification are determined.
- System and software design: Overall architecture of the system is established. Programming language are decided , database and high level technical details are planned.
- Implementation and unit testing : Software design is implemented in the set of programs or program units. Unit testing involves verifying that each unit meets its specification. Basically coding starts
- Integration and system testing: program units are integrated and tested as a complete system. After testing, software system is delivered to the customer.

A. Waterfall Model

- **Operation and maintenance: This is the longest life cycle phase. System is installed and put into practical use.**
- **Maintenance involves correcting errors and enhancing the system services.**
- **The following phase should not start until the previous phase has finished.**

Waterfall model : Can be used when -

- Requirements are not changing frequently
- Application is not complicated and big
- Project is short
- Requirement is clear
- Environment is stable
- Technology and tools used are not dynamic and is stable
- Resources are available and trained

Advantage and Disadvantage

- **Advantage:**
- Before next phase , each phase must be completed
- Best for the project when requirement are well defined
- Verification and validation must be completed for each stage after completion of each stage

- **Disadvantage**:
- Error can be fixed only during the phase
- Not desirable for complex project
- Documentation occupies a lot of time for developers and testers
- Clients feedback cannot be included during development phase

B. Incremental Development

- Basic Concept is developing an initial implementation, getting feedback from users and others, and evolving the software in different versions.
- Most common approach of development of application systems
- This approach can be plan - driven, agile or, combination of many approaches
- In a plan driven approach, system increments are identified in advance
- In agile the early increments are identified, but the development of later increments depends on progress and customer priorities.

B. Incremental Development

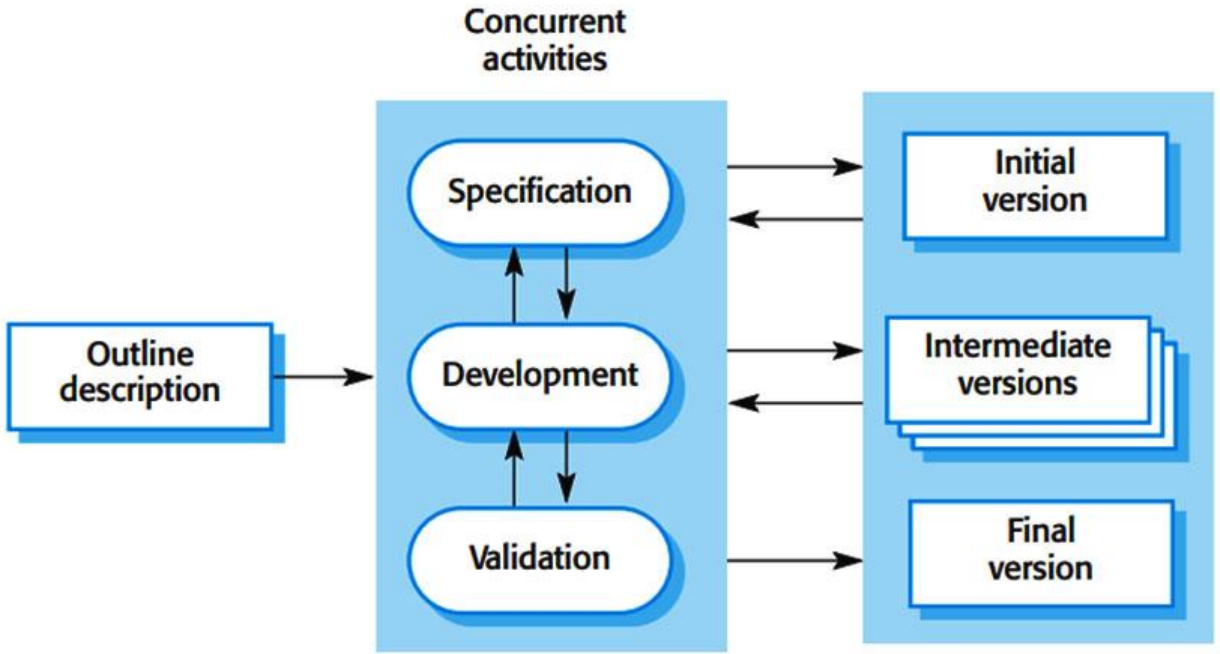


Figure 2.2 Incremental development

Advantage of Incremental model over Waterfall

- The cost of implementing requirements changes is reduced.
- The amount of analysis and documentation that has to be redone is significantly less than is required with the waterfall model.
- It is easier to get customer feedback on the development work that has been done.
- Customers can comment on demonstrations of the software and see how much has been implemented.
- Early delivery and deployment of useful software to the customer is possible, even if all of the functionality has not been included.
- Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

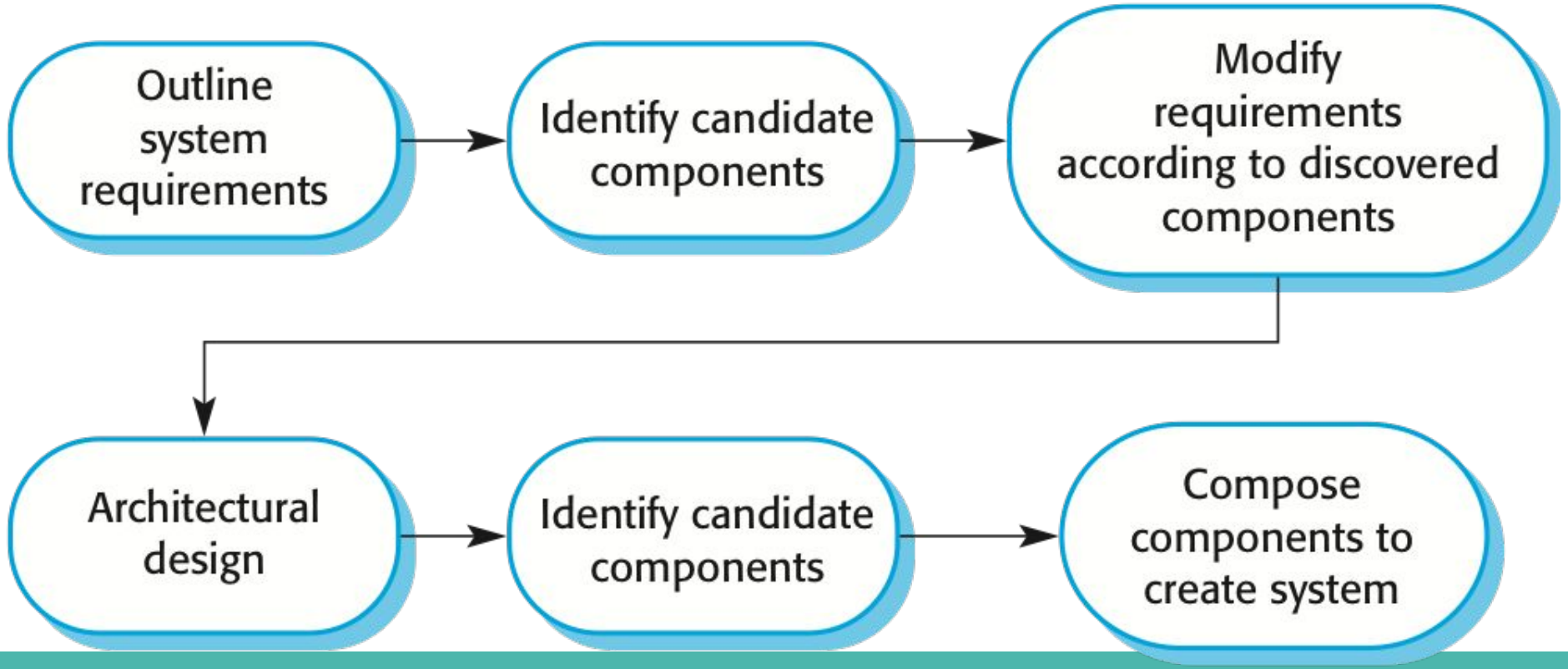
Problem In Incremental model

- The process is not visible.
- Managers need regular deliverables to measure progress.
- If systems are developed quickly, it is not cost effective to produce documents that reflect every version of the system
- System structure tends to degrade as new increments are added.
- Regular change leads to messy code as new functionality is added in whatever way is possible.

C. Component Based Software Engineering

- Development based on creation and reuse of software components
- Components are modular, self contained units that can be assembled to create large systems
- No need to write code from scratch, directly use components
- Follows philosophy "*buy, don't built*"
- Characteristics :
  - Modular: self contained
  - Reusable
  - Interoperable: able to be exchanged
  - Composable

Component Based Software Engineering



Why CBSE?

- Increase Quality, evolvability and maintainability
- Increase productivity
- Shortens Development time
- Easy to assemble and less costly to build the system
- Use assembled approach

CBSE Framework Activities:

- Component Qualification
- Component Adaptation
- Component composition
- Component update

What is Component?

- **Independent and replicable part of a system that fulfills a clear function.**
- **Executable entity which can be made up of one or more executable objects**
- **Deployable**
- **Documented**

CBSE : Component Qualification

- It ensures that a candidate component will perform the function required
- Component must fit into the arhitectural specified for system
- Other factors included inside qualification are :
  - Application program interface(API)
  - Runtime requirements
  - Network protocols
  - Os interface etc.

CBSE : Component Adaptation

- After the component is qualified then also conflict may arise in some areas.
- To avoid conflicts an adaptation technique called component wrapping is often used
- White box Wrapping : Integration conflicts are removed by making code level modifications to the code
- Grey - Box Wrapping: Used when component uses API or extension language
- Black - Box Wrapping: Related to the functionality of components

CBSE: Component Composition

- It involves the assembling of qualified, adapted and engineered components
- It involves :
- Data Exchange model: Drag and drop mechanism to transfer the contents
- Automation: Tools, scripts should be implemented to facilitate interaction between reusable components
- Structured storage: Heterogeneous data should be organized
- Underlying object model: components developed in different languages

Advantages / Disadvantages

- **Advantage:**
  - Reduce development time
  - Increase Productivity
  - Reduce cost
  - Reliability increases
  - Complexity management and flexibility

- **Disadvantages:**
  - Development of components
  - Quality of component is questionable
  - Maintenance cost

Evolutionary Process Model

- Software evolves over a period of time, in the same way business and product requirement often change as development proceeds.
- Making a comprehensive complete software at a single time is impossible
- In these and similar situations, you need a process model that has been explicitly designed to accommodate a product that evolves over time.
- Evolutionary models are iterative. They are characterized in a manner that enables you to develop increasingly more complete versions of the software.

Evolutionary Process Model - Types

- **Two common evolutionary process models are:**
  - **Prototyping**
  - **The Spiral model**

"Plan to throw one away. You will do that, anyway. Your only choice is whether to try to sell the throwaway to customers."

Frederick P. Brooks

Prototyping Model

- When your customer has a legitimate need, but is clueless about the details, develop a prototype as a first step
- The developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system, or the form that human-machine interaction should take.
- For all these situations prototype model is best
- It consists of 5 stages:

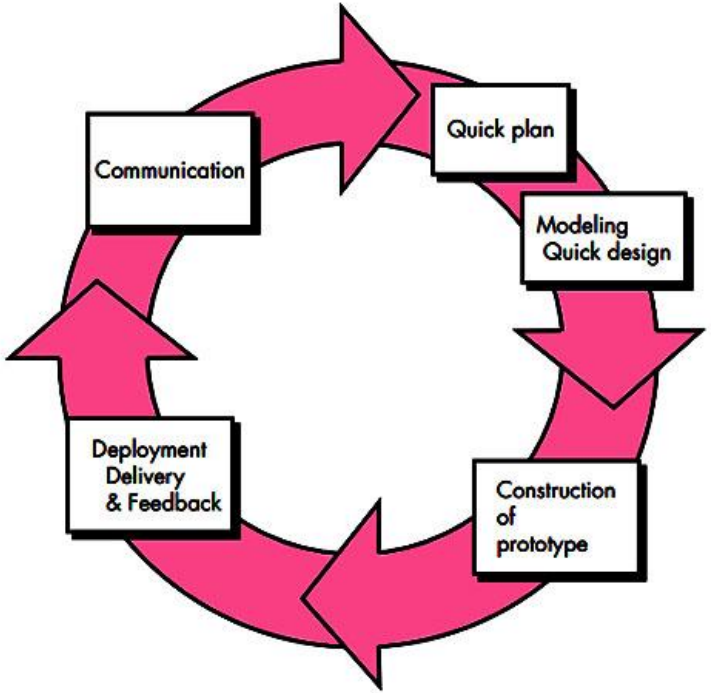
- Communication
- Quick Plan
- Modeling quick design,

- Construction of prototype
- Deployment delivery and feedback

Prototyping Model

FIGURE 2.6

The prototyping paradigm



Prototyping Model

- **Communication : You meet with other stakeholders to define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory.**
- **Quick Plan: A prototyping iteration is planned quickly, and modeling (in the form of a "quick design") occurs.**
- **Modelling Quick Design: A quick design focuses on a representation of those aspects of the software that will be visible to end users (e.g., human interface layout or output display formats)**

Prototyping Model

- **Construction of Prototype: Based on the design, A simple version of the software is developed**
- **Deployment, Delivery and feedback: The prototype is deployed and evaluated by stakeholders, who provide feedback.**
- **Iteration occurs as the prototype is tuned to satisfy the needs of various stakeholders, while at the same time enabling you to better understand what needs to be done.**
- **In most projects, the first system built is barely usable. It may be too slow, too big, awkward in use or all three. There is no alternative but to start again, smarting but smarter, and build a redesigned version in which these problems are solved**

Why not Prototyping?

- For Stakeholders:
- Stakeholders take prototype as the complete working system, they are unaware that the prototype is held together haphazardly.
- When informed that the product must be rebuilt so that high levels of quality can be maintained, stakeholder might demand that “a few fixes” be applied to make the prototype a working product

Why not Prototyping?

- **For Software Engineer:**
- **As a software engineer, you often make implementation compromises(Taking a shortcut) in order to get a prototype working quickly.**
- **An inappropriate operating system or programming language may be used**
- **An inefficient algorithm may be implemented simply to demonstrate capability.**
- **After a time, you may become comfortable with these choices and forget all the reasons why they were inappropriate. The less-than-ideal choice has now become an integral part of the system.**

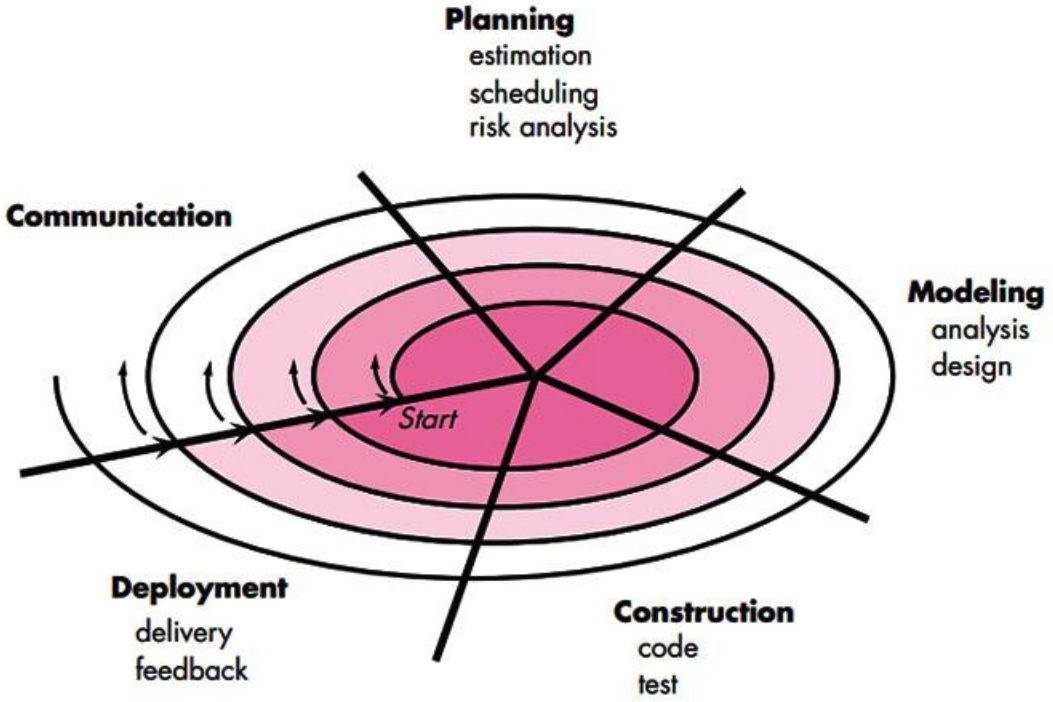
The Spiral Model

- It can be said as a combination of iterative nature of prototyping and controlled and systematic approach of waterfall model.
- It supports rapid development and results in complete version of the software.
- During early iterations, the release might be a model or prototype. During later iterations, increasingly more complete versions of the engineered system are produced.

The Spiral Model

FIGURE 2.7

A typical spiral model



Spiral Model : Phases

- **Planning:** The first phase of the Spiral Model is the planning phase, where the scope of the project is determined and a plan is created for the next iteration of the spiral.
- **Risk Analysis:** In the risk analysis phase, the risks associated with the project are identified and evaluated.
- **Engineering:** In the engineering phase, the software is developed based on the requirements gathered in the previous iteration.

Spiral Model : Phases

- **Evaluation:** In the evaluation phase, the software is evaluated to determine if it meets the customer's requirements and if it is of high quality.
- **Planning:** The next iteration of the spiral begins with a new planning phase, based on the results of the evaluation.

Spiral Model : Four Quadrant

- **Objectives Determination and identify alternative solutions:**
- **Requirements gathered and objectives are identified**
- **Alternative solutions are proposed**

- **Identify and resolve Risks:**
- **Possible solutions are evaluated**
- **Risks are identified and resolved**
- **At the end of this quadrant prototype is build**

Spiral Model : Four Quadrant

- **Develop next version of product:**
- **Testing and new features are added**
- **Next version of software is available**

- **Review and plan for next phase:**
- **Customer evaluate the latest version and planning for next phase is started**

Advantage and Disadvantage

- Advantages :
- Risk Handling: Risk analysis in every phase
- Good for large projects
- Flexibility in requirements: change can be done at any time
- Customers Satisfaction
- Incremental and iterative approach
- Improved Communication
- Improved quality

- Disadvantages :
- Complex
- Expensive
- Too much dependability on risk analysis
- Difficulty in time management

Rapid Software Development

- **Agile** Methods
- **Extreme** Programming
- **Rapid** Application Development

Agile Method

- **Incremental and iterative process**
- **Every iteration is of short time frame which mostly last from two to four weeks**
- **Each build is incremental in terms of functionality**

Agile Development

Fig. Agile Model

Agile Characteristics

- **Modularity**
- **Iterative**
- **Time - bound**
- **Incremental**
- **People Oriented**
- **Collaborative**
- **Motivating the team**

Agile Manifesto

- **Individuals and interactions** over **processes and tools**
- **Working Software** over **comprehensive documentation**
- **Customer collaboration** over **Contract Negotiation**
- **Responding to change** over **following a plan**

Agility Principle

- Our highest priority is to satisfy the customer through *early and continuous* delivery of valuable software.
- *Welcome changing requirements*, even late in development.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and Developers must work together daily throughout the project.

Agility Principle

- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.

Principles Of Agile

- Customer Satisfaction
- Working Software
- Measure of Progress
- Late Changes are welcome
- Face to Face Communication
- Technical Excellence
- Simplicity

When Agile Development ..?

- It is equally difficult to predict how customer priorities will change as the project proceeds
- For many types of software, design and construction are interleaved. That is, both activities should be performed in together so that design models are proven as they are created.
- It is difficult to predict how much design is necessary before construction is used to prove the design.
- Analysis, design, construction, and testing are not as predictable
- For all these assumptions agile is the best solution

Phases Of Agile Development:

- Requirement Gathering
- Design the requirement
- Develop/ Iteration
- Test
- Deployment
- Feedback

- ***Compared to Waterfall, Agile cycles are short. There may be many such cycles in a project. The phases are repeated until the product is delivered.***

Human Factor in Agile Development

- **Competence:** Individual must have skills and knowledge required for the product development
- **Common focus :** Must have one goal
- **Collaboration :** Proper communication
- Decision making ability
- Problem solving ability
- Mutual trust and respect

When To use Agile Model

- Frequent changes that need to be implemented.
- Projects without any existing process
- Projects where the product owner is highly accessible
- Projects with flexible timelines and budget

Advantage/ Disadvantage

- **Advantages :**
- Communication one to one with clients
- Provides realistic approach of software development
- Updated version of software are released every week
- Delivers early partial working solution
- Changes are acceptable at any time

- **Disadvantages:**
- Not a suitable method for complex dependencies
- Without clear information from customer team can be misled
- Documentation and design ar not given much attention

Extreme Programming(xp)

- An approach of agile model
- Lightweight Agile Process
- Instead of lots of documentation XP focusses on plenty of feedback
- Embrace change: iterate often, design and redesign, code and test frequently, keep the customer involved
- Deliver product in short time (2 weeks)

Core Values Of XP

- **Communication :**
- **Implement practices that force communication in a positive way.**
- **Simplicity:**
- **Develop simplest product that meets customer needs**
- **Feedback:**
- **Courage:**
- **Be prepared to make hard decisions that support other principles**

Requirement Scenarios

- In XP, user is also a part of the XP team
- User requirements are expressed in the form of use case or user story
- Use case are then broken into implementation task
- These task are the basis of the schedule and the cost estimates

Prescribing Medication Story

Prescribing medication

The record of the patient must be open for input. Click on the medication field and select either 'current medication', 'new medication' or 'formulary'.

If you select 'current medication', you will be asked to check the dose; If you wish to change the dose, enter the new dose then confirm the prescription.

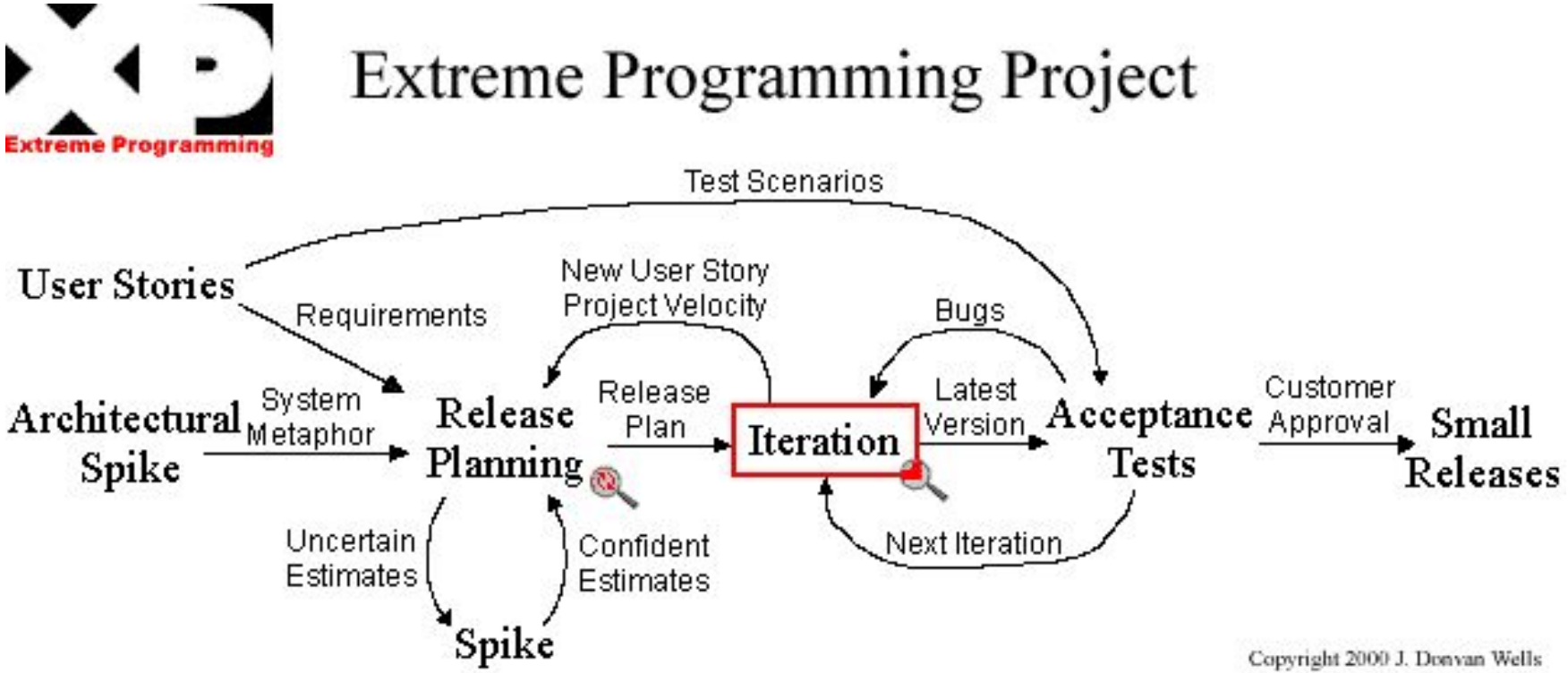
If you choose, 'new medication', the system assumes that you know which medication you wish to prescribe. Type the first few letters of the drug name. You will then see a list of possible drugs starting with these letters. Choose the required medication. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

If you choose 'formulary', you will be presented with a search box for the approved formulary. Search for the drug required then select it. You will then be asked to check that the medication you have selected is correct. Enter the dose then confirm the prescription.

In all cases, the system will check that the dose is within the approved range and will ask you to change it if it is outside the range of recommended doses.

After you have confirmed the prescription, it will be displayed for checking. Either click 'OK' or 'Change'. If you click 'OK', your prescription will be recorded on the audit database. If you click 'Change', you reenter the 'Prescribing medication' process.

XP Overview


Extreme Programming
Extreme Programming Project
Test Scenarios
User Stories
Requirements
New User Story
Project Velocity
Bugs
Architectural Spike
System Metaphor
Release Planning
Release Plan
Iteration
Latest Version
Acceptance Tests
Customer Approval
Small Releases
Uncertain Estimates
Confident Estimates
Spike
Next Iteration
Copyright 2000 J. Donvan Wells

Advantages / Disadvantages

- **Advantages:**
- **Large Projects** are divided into manageable amounts
- Reduce **time and cost**
- **Saves a lot of time** because they don’t use too much documentation
- Simplicity
- Reduces the risk

- **Disadvantages:**
- Focus on the **code** rather than on design
- Requires a **detailed planning** from start
- Doesn’t measure quality assurance

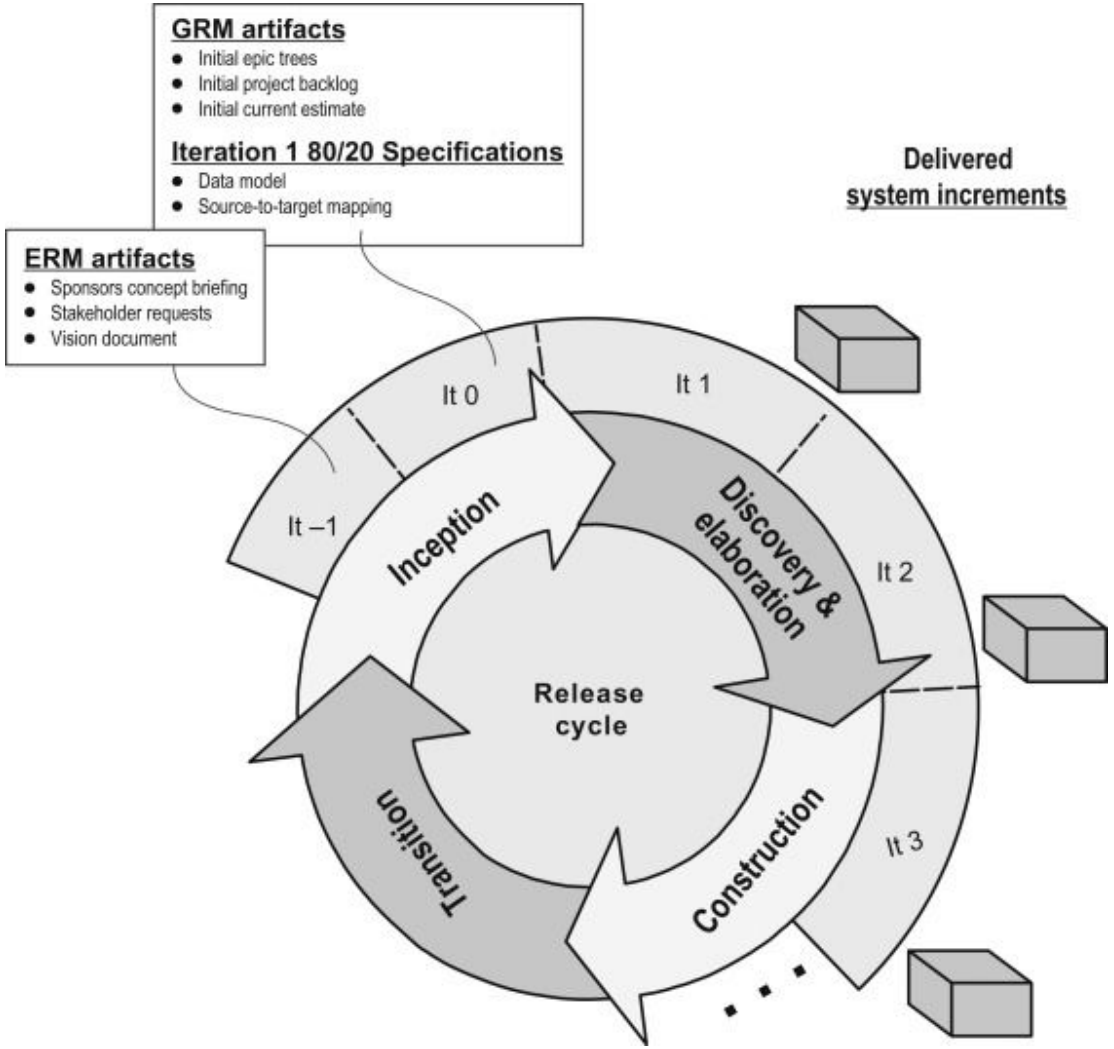
Software Prototyping

- **Rational Unified Process(RUP)**
- **Computer Aided Software Engineering(CASE):**
- **Tools**

Rational Unified Process

- **Unified Process:** An iterative and incremental software development process framework for building software systems
- Iterative and evolutionary approach always start with
  - Incomplete and imperfect knowledge
- Iterative and evolutionary have the following advantage:
  - Effective management of changing requirements
  - Continuous integration of changes
  - Early understanding of the system
  - Ongoing risk assessment

Rational Unified Process



Rational Unified process

- Refinement of unified process
- Provides a disciplined approach to assigning tasks and responsibilities with a development organization
- Goal - To ensure the production of high quality software with:
  - User satisfaction
  - Within schedule and budget

Rational Unified Process: Best Practices

- Develop software iteratively
  - Best in case:
    - Difficult to define the problem upfront
    - Design the entire solution at once
  - Each iteration ends with a release
- Manage Requirements
  - Use case diagrams(UML) for capturing functional requirements
  - Can be traced
- Visually model software
- Use component based architecture

Rational Unified Process: Phase

- Inception Phase
  - Vision Document,
    - Scope of the system, risk, cost etc.
- Elaboration Phase
  - Risk identification, problem domain, analysis and architecture
- Construction Phase
  - Build the software, can be broken into iterations
- Transition Phase
  - Transition from development to production

Computer Aided Software Engineering (CASE)

- To speed up the software system building process, a new concept of designing software is introduced in the 70's, called as Computer Aided Software Engineering.
- CASE is the use of software tools to help in the development and maintenance of software.
- CASE provides the Automated support for software process activities which means using one software to build other.

Why CASE Tools ?

- Time saving and reducing coding and testing time
- Enrich graphical techniques and data flow
- Enhanced analysis and design development
- Create and manipulate documentation
- Speed of development increased

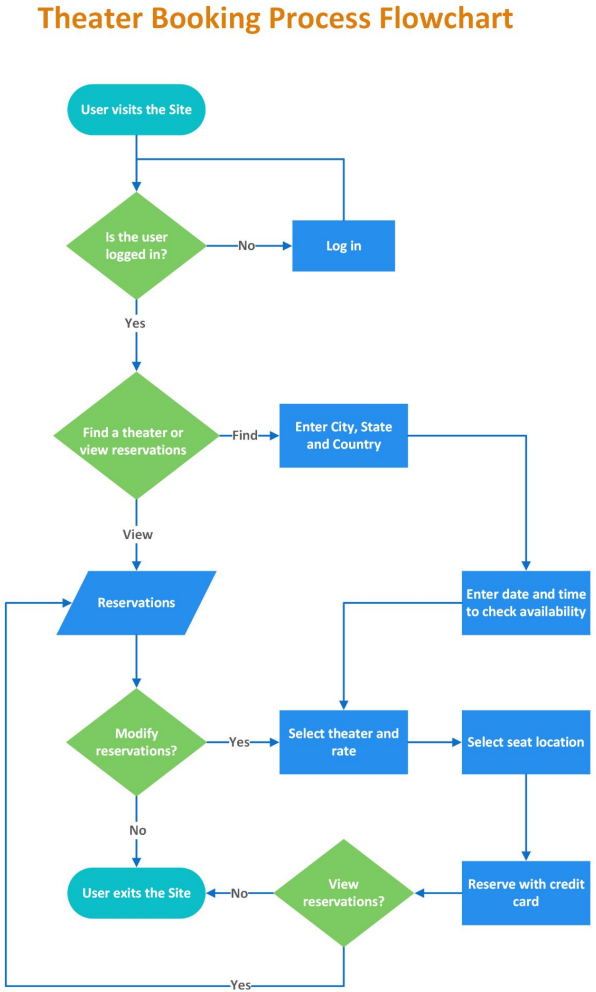
CASE Tools

- Diagram Tools
- Project management Tools
- Documentation Tools
- Web Development Tools
- Quality Assurance Tools
- Maintenance Tools

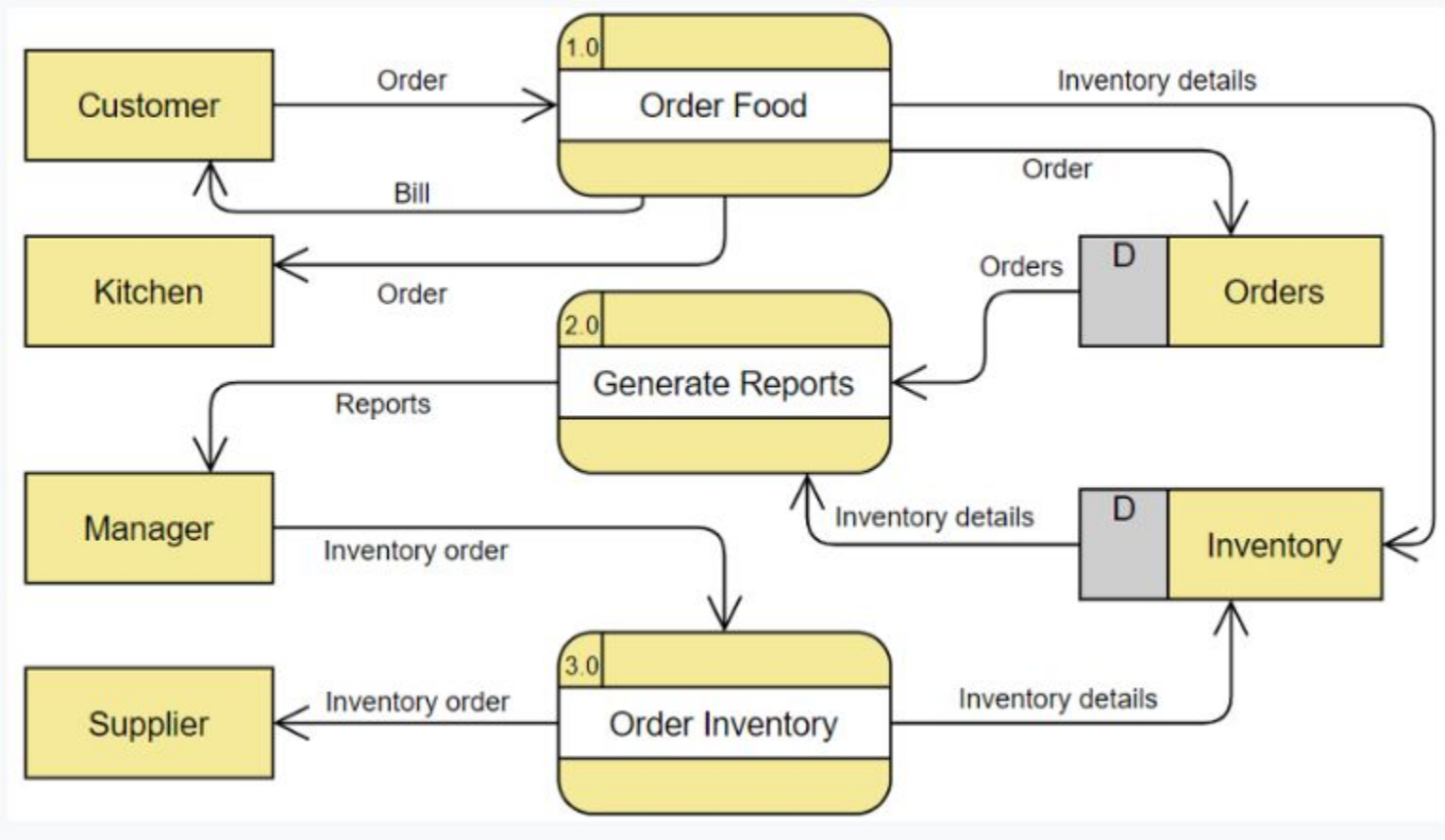
Diagram Tools:

- These tools are used to represent system components, data and control flow among various software components.
- Represent system structure in graphical form
- Examples:
  - Flow Chart Marker Tool (Smartdraw.com)
  - DFD's (Data Flow Diagram)
  - ERD's (Entity Relationship Diagram)

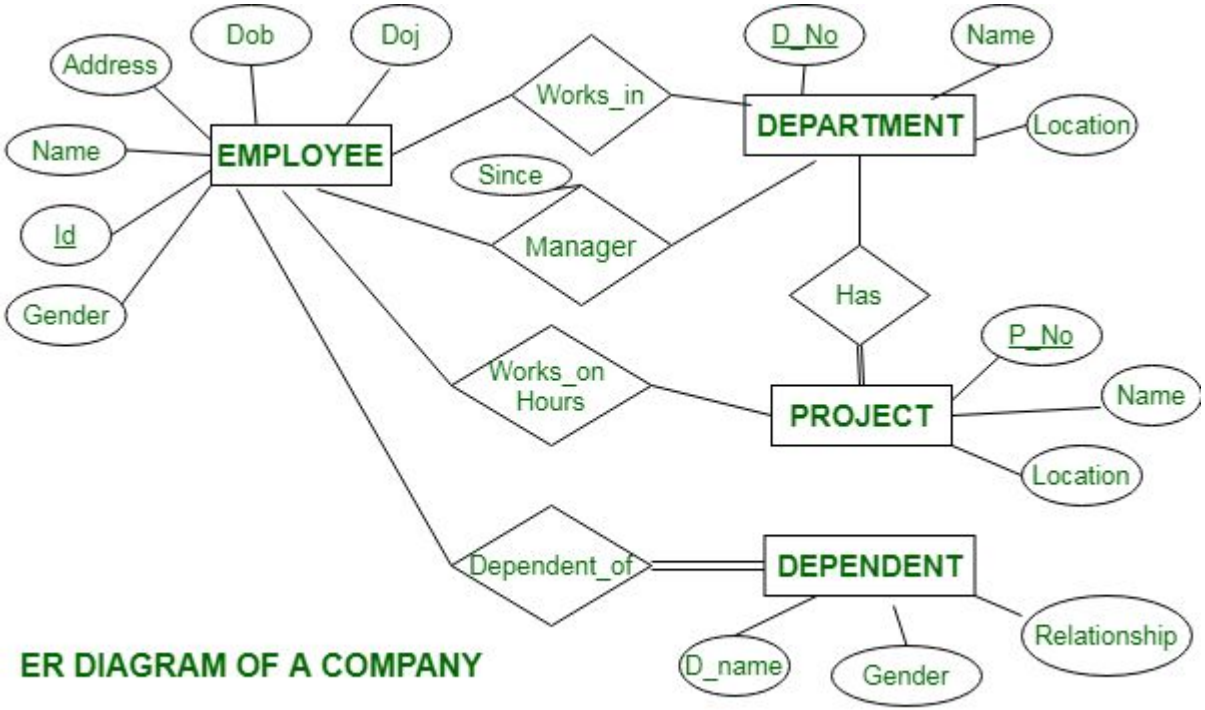
Flowchart


Theater Booking Process Flowchart
User visits the Site
Is the user logged in?
No
Log in
Yes
Find a theater or view reservations
Find
Enter City, State and Country
View
Reservations
Enter date and time to check availability
Modify reservations?
Yes
Select theater and rate
Select seat location
No
User exits the Site
No
View reservations?
Reserve with credit card
Yes

DFD


Customer
Order
1.0
Order Food
Inventory details
Bill
Order
Kitchen
Order
D
Orders
Orders
2.0
Generate Reports
Reports
Manager
Inventory details
D
Inventory
Inventory order
3.0
Supplier
Inventory order
Order Inventory
Inventory details

ER DIAGRAM


Address
Dob
Doj
Name
Id
Gender
EMPLOYEE
Works_in
D_No
Name
DEPARTMENT
Location
Since
Manager
Has
Works_on
Hours
P_No
PROJECT
Name
Location
Dependent_of
DEPENDENT
Relationship
D_name
Gender
ER DIAGRAM OF A COMPANY

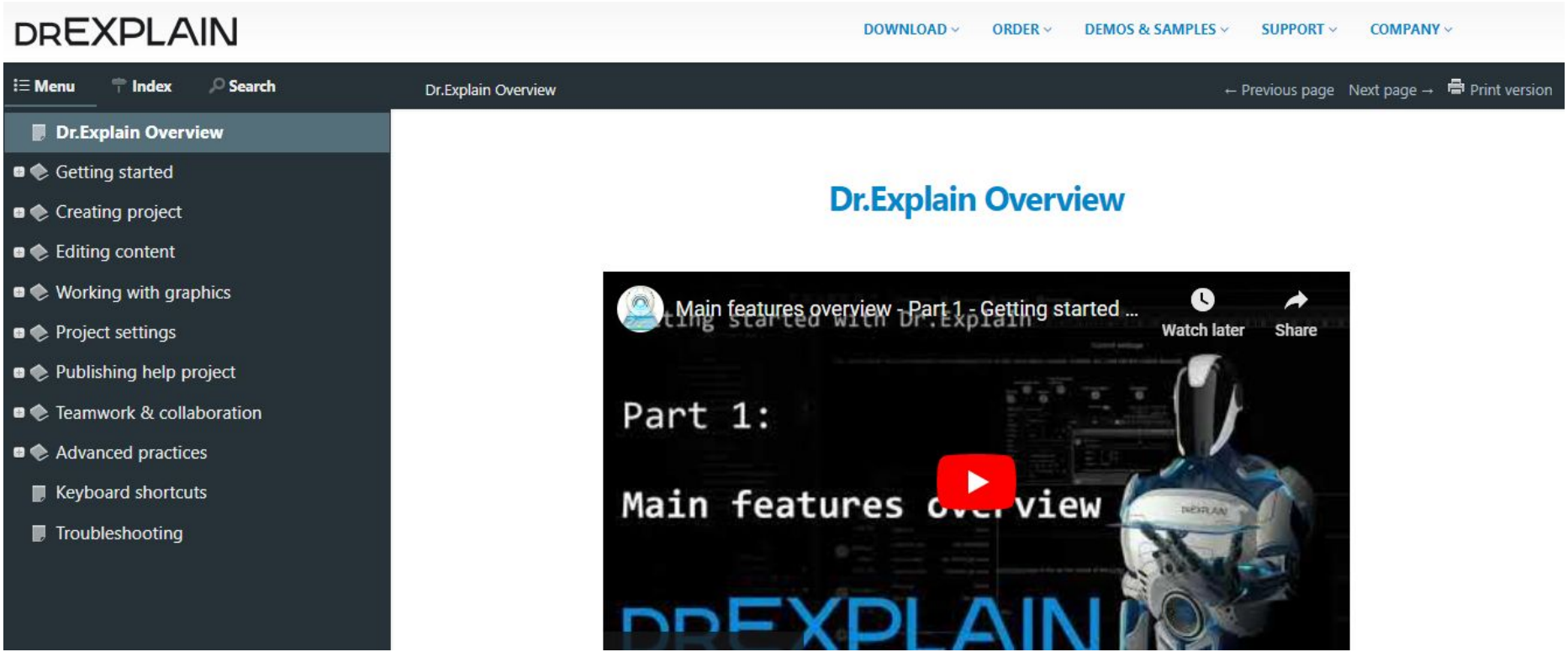
Project Management Tools:

- **Used For :**
  - **Project Planning and Scheduling**
  - **Cost and effort Estimation**
  - **Resource Planning**
  - **Example:**
    - **Creative Pro Office**

Documentation Tools:

- Helps to generate documents for technical users and end users
- Training manuals, Installation Manual, user manual can also be generated by documentation tools
- Example:
  - DrExplain

DrExplain


DREXPLAIN
DOWNLOAD
ORDER
DEMOS & SAMPLES
SUPPORT
COMPANY
Menu
Index
Search
Dr.Explain Overview
← Previous page
Next page →
Print version
Dr.Explain Overview
Getting started
Creating project
Editing content
Working with graphics
Project settings
Publishing help project
Teamwork & collaboration
Advanced practices
Keyboard shortcuts
Troubleshooting
Dr.Explain Overview
Main features overview - Part 1 - Getting started ...
Watch later
Share
Part 1:
Main features overview

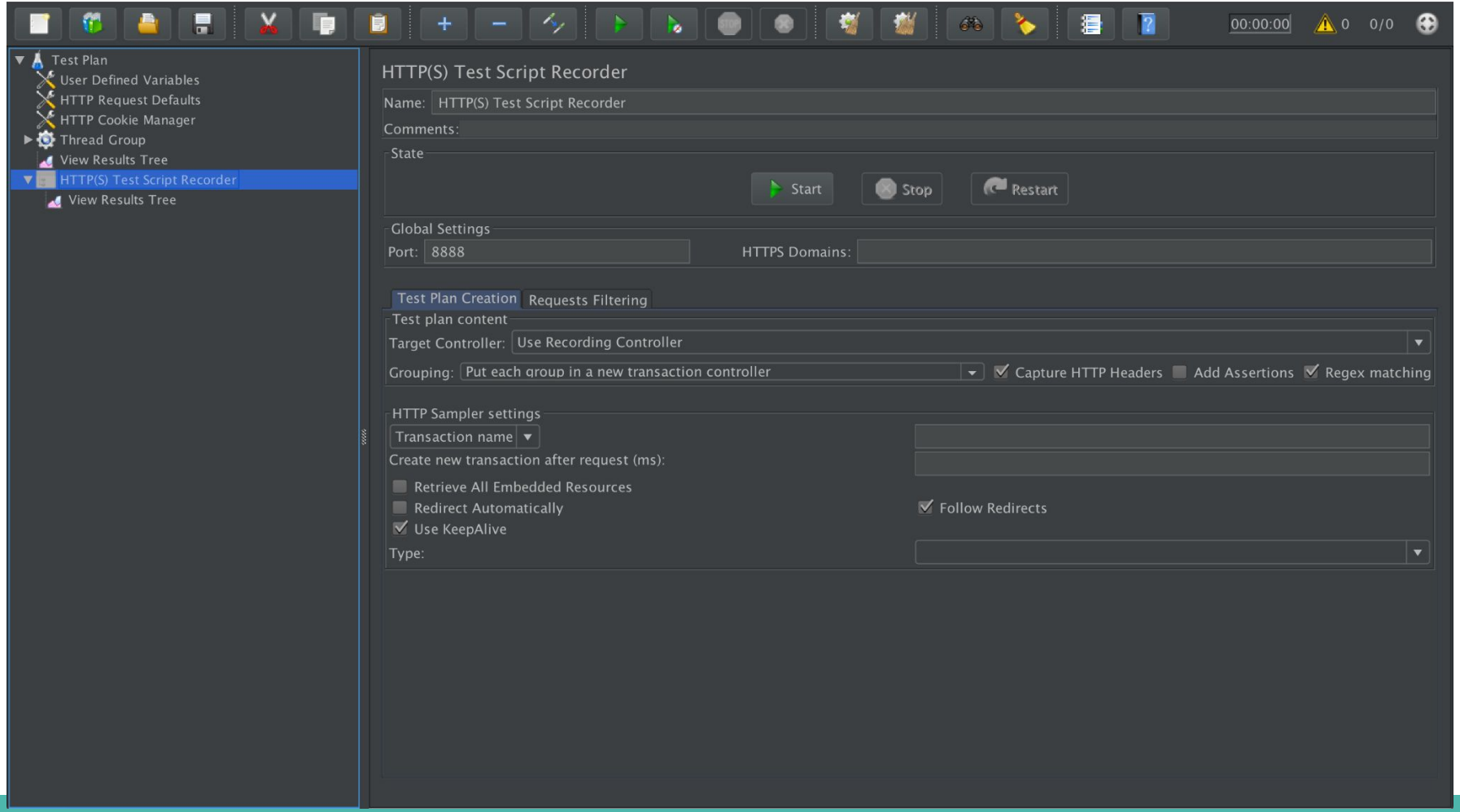
Web Development Tools

- **Helps in designing web pages with all elements available like:**
- **Forms, text, script, graphic and so on.**
- **Web tool also provide live preview of what is being developed and how it looks after completion**
- **Example**
  - **Adobe Edge Inspect**

Adobe Edge Inspect

Quality Assurance Tools

- **Helps to monitor Process and methods adopted to develop the software product in order to ensure quality requirements**
- **Example:**
  - **JMeter**


Test Plan
User Defined Variables
HTTP Request Defaults
HTTP Cookie Manager
Thread Group
View Results Tree
HTTP(S) Test Script Recorder
View Results Tree
00:00:00
0
0/0
HTTP(S) Test Script Recorder
Name: HTTP(S) Test Script Recorder
Comments:
State
Start
Stop
Restart
Global Settings
Port: 8888
HTTPS Domains:
Test Plan Creation
Requests Filtering
Test plan content
Target Controller: Use Recording Controller
Grouping: Put each group in a new transaction controller
Capture HTTP Headers
Add Assertions
Regex matching
HTTP Sampler settings
Transaction name
Create new transaction after request (ms):
Retrieve All Embedded Resources
Redirect Automatically
Follow Redirects
Use KeepAlive
Type:

Maintenance Tools

- **Helps in modification in software product after it is delivered**
- **Example:**
  - **Bugzilla**

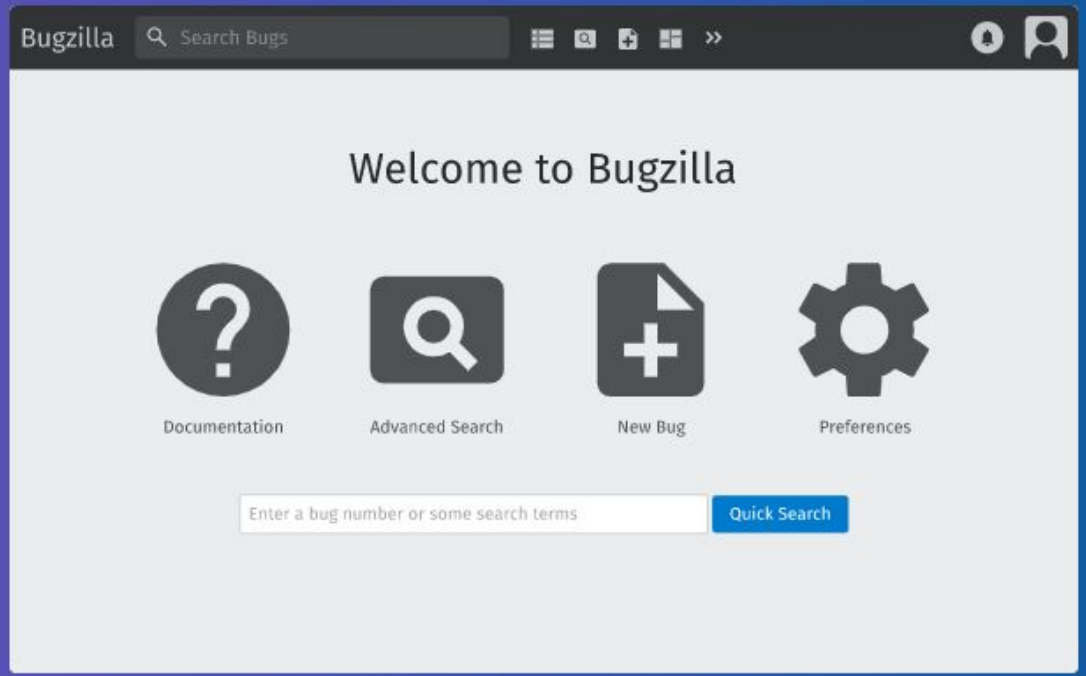
Bugzilla

About ▾ News ▾ Download Support Contribute ▾ Search

The software solution designed to drive software development

Bugzilla lets you plan, organize and release software on your own teams' schedule.

Learn more » Try it out »



Advantages / Disadvantage

- **Advantages:**
- **Improves Quality and Productivity**
- **Produces system that is more closer towards user need and requirement**
- **Excellent Documentation**
- **Flexible System**
- **Reduce time for error correction and maintenance**

- **Disadvantages:**
- **Complex**
- **Quality tools are expensive**
- **Require training**
- **Difficult to use with existing system**

Assignment - II

- What are the key challenges that software Engineering Face during software development ? Explain.
- What is software model? List the types of software model.
- Explain agile methods and software prototyping
- What are the types of software requirements? Explain functional, non functional, domain and user requirements
- What are the skills necessary to handle software project?
- What is waterfall model? Describe the activities of waterfall model and also mention its drawbacks.

Assignment - III

- **What is CASE? Explain the importance of CASE tools in software development life cycle.**
- **What is programming language? Explain different software development tools.**
- **Differentiate between software engineering and computer science**
- **Explain spiral and prototyping model**
- **Compare agile software development with prototyping model**

Literature Review

- **Review the literature and write a review paper based on that literature.**
- **Different process model**
- https://d1wqtxts1xzle7.cloudfront.net/74890408/download-libre.pdf?1637333790=&response-content-disposition=inline%3B+filename%3DUnderstanding_why_in_software_process_mo.pdf&Expires=1680159142&Signature=ghfmqHMBX2c71kLBGLFPKo2Vrn~m4XQ4JZgKLqa5rG645Ce5ICbjVISqJ-fVgCgnpjiikkBiwrTeHQUhqWxTwGLD4okD6ApNlnHB6lL3GONH6btyd~IYxLB-kDHANeDjrqKJtAH5GK9y40BmgOdYxiOi61rBbXmkRddqn5zjJ1GyBzeSZf8nkGr1I3NQcliEUI4bmiYFrXAzFME-1I4Bexn2~BdOpP3ID5iik0taELLhf0dmTuLDZFUFnDsd243ddFQpnG9~iDcPbkTNHws0Fx2~TG54UiYKp-GeTIJZ38dPw9vM5maCAFBuIUjC~-YSNVuH5jmTS9EwFnb6TWGL-Q__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA
- **Agile Software model**
- https://www.cse.msu.edu/~cse435/Homework/HW3/p3-aoyama.pdf