Introduction to Software Engineering

At the end of this course, you will be able to:

- Understand what Software Engineering is and why it is important
- Introduce the software process and different types of models
- Understand the importance and usage of DFDs
- Understand what is software quality measurements and the activities involved in the improvement of quality
- Answer some fundamental questions about Software Engineering

Software : A definition

"Software systems are perhaps the most intricate and complex….of the things humanity makes."

-Fred Brooks

Software is

- Intangible
- Complex
- Flexible

Importance of Software

- Software can have a huge impact in any aspect of society.
- Where exactly you can find software?

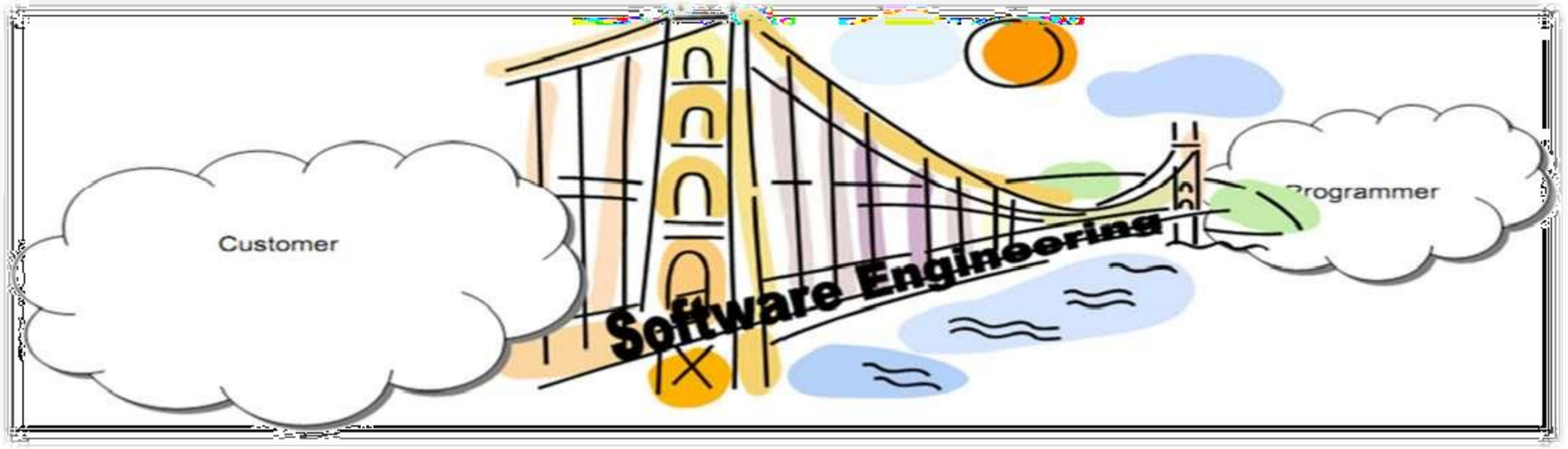
Software Types

- There are normally two types of Software
  - Generic
  - Customized

- Application types
  - Standalone
  - Interactive
  - Embedded
  - Batch Processing Systems
  - Data Collection Systems

What is the role of software engineering?

- To capture the customer's business needs and specify the "blueprints" for the system so that the programmers can implement it.



Problems in software development

- Developing large/complex software application is very challenging
- Effort intensive
- High Cost
- Long development time
- Changing needs for users
- High risk of failures

Successful software systems

- Software development projects have not always been successful
- When do we consider a software application successful?
- Development completed
  - It is useful
  - It is usable
  - It is used

Common issues

- The final software doesn’t fulfill the needs of the customer
- Poor quality of software
- Hard to extend and improve
- Bad documentation
- More time and costs than expected

Solution?

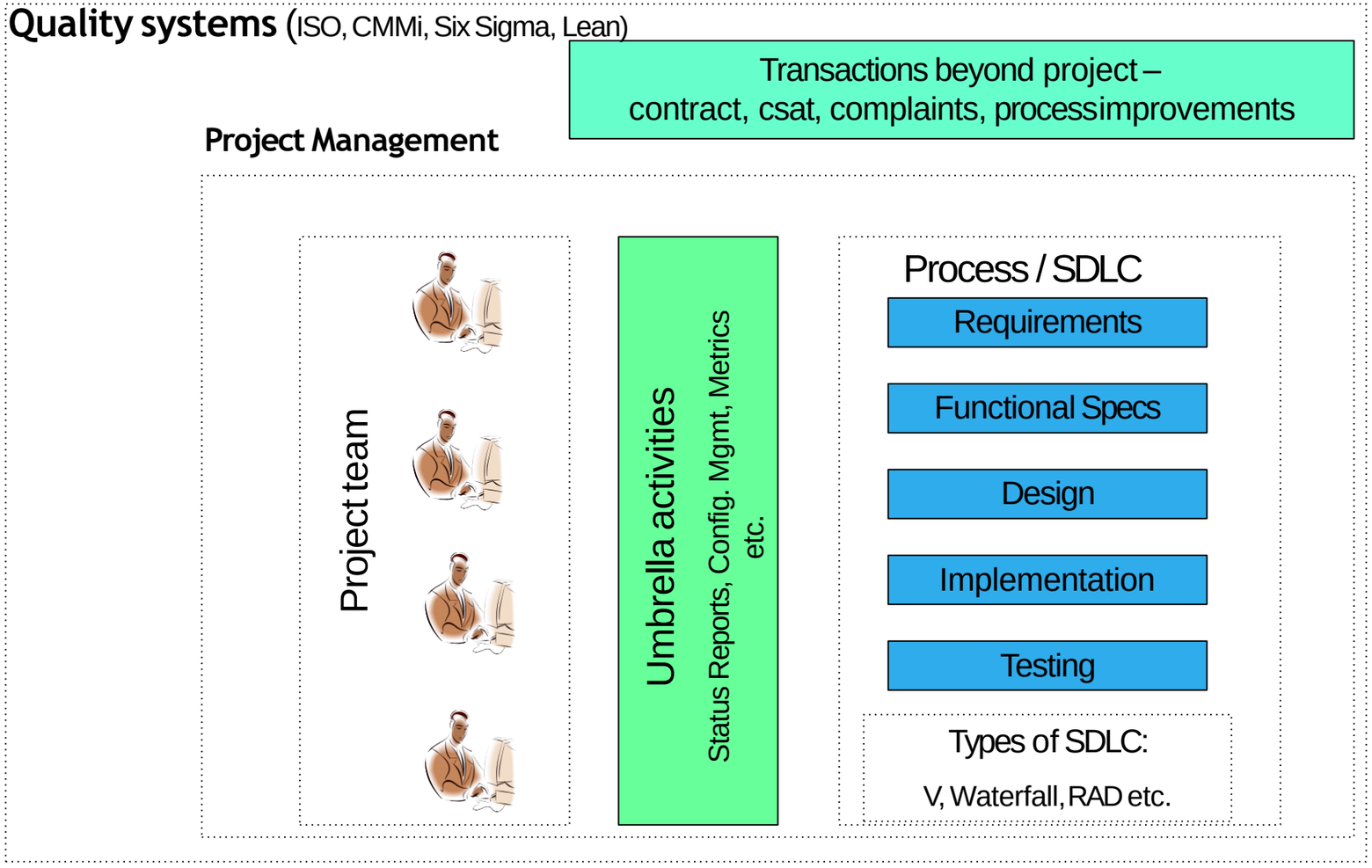
Software Engineering

- An engineering discipline that applies theories, methods, and tools to solve problems related to software production and maintenance.
- Software engineers strive to deliver high-quality software, on time and within budgets.

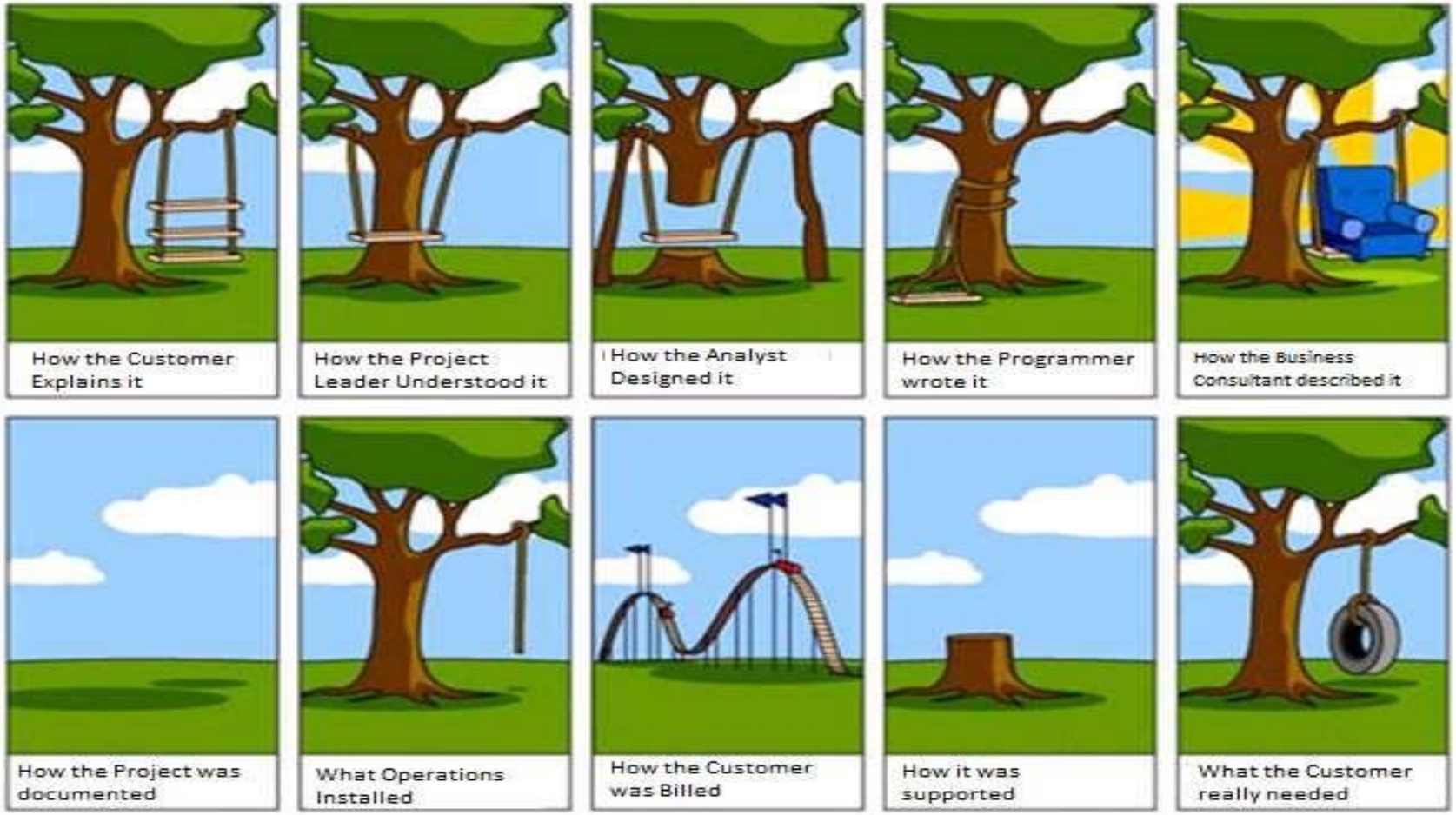
Software Engineering

Every Project to concentrate On

1. Quality Aspects
2. Project Management



Why Design System?



Why is Software engineering important?

- Financial, Security, and Safety critical systems rely on software.
- Software mediates every aspect of our internet experience.
- The economics of all developed nations are dependent on software.
- An increasing need to develop high quality software in a cost effective manner.

What is Software Process?

A set of activities and their output, which results in a software product. Four fundamental process activities:

- Specification: Defines what the software should do, and its operational constraints
- Design and implementation: Designs the solutions, and produces the source code to meet the specification
- Validation: Checks that the software produced is what the customer wants
- Evolution: Changes made to the software that meet user's changing needs

What is Software Requirement Specification?

- The [SRS] specifies the requirements … and the methods to be used to ensure that each requirement has been met.
- It is a document that you prepare:
  - after customer gives you their system specifications
  - before you design the system
- Purpose
  1. “It provides feedback to the customer.”
  2. “It decomposes the problem into component parts.”
  3. “It serves as an input to the design specification.”
  4. “It serves as a product validation check.”

- Functional requirements
- Nonfunctional requirements
- It doesn't provide
- Design suggestions
- Possible solutions to technology or business issues
- Any other information other than what the development team understands the customer's system requirements to be

- Again from the IEEE standard:
- The basic issues that the SRS writer(s) shall address are the following:
  - Functionality - What is the software supposed to do?
  - External interfaces - How does the software interact with people, the system's hardware, other hardware, and other software?
  - Performance -What is the speed, availability, response time, recovery time of various software functions, etc.?
  - Attributes-What are the portability, correctness, maintainability, security, etc. considerations?
  - Design constraints imposed on an implementation - Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) etc.?

INTRODUCTION TO SOFTWARE ENGINEERING > IEEE Standard 830-1998 Characteristics of a good SRS

- Correct
- Unambiguous
- Complete
- Consistent
- Ranked for importance and/or stability
- Verifiable
- Modifiable
- Traceable

- Correct: every requirement given in SRS is a requirement of the software
- Unambiguous: every requirement has exactly one interpretation
- Complete: includes all functional, performance, design, external interface requirements; definition of the response of the software to all inputs (valid and invalid)
- Consistent: internal consistency

Generic software process model

- Waterfall model
- Evolutionary models
  - Prototyping model
  - Spiral model
- Increment model
  - Iterative model
  - RAD model
- Component based model

Waterfall Model

- Classical life cycle and linear sequential model.
- One of the oldest model and is widely used.
- Suggests a systematic sequential approach from requirement analysis up to support phase.

Phases of waterfall model

- Requirement analysis and definition
  - The goals and constraints of the system to be developed are established in consultation with system users
  - They are then defined in detail and serve as a system specification
- System and software design
  - Partitions the requirements to either hardware or software
  - Software design involves identification of necessary fundamentals of the software requirement and their relationship with those systems

Phases of waterfall model

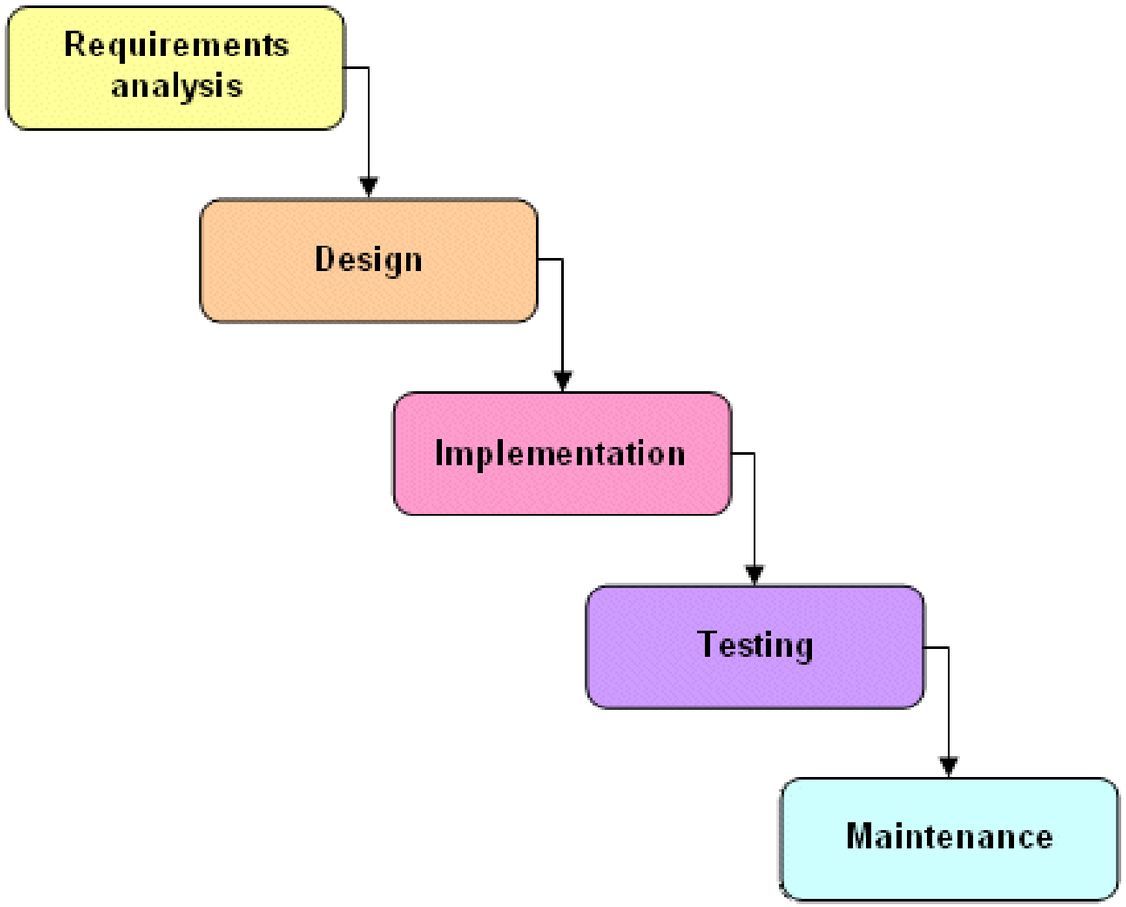
- Implementation
  - During this stage the whole of software components are integrated as a set of programs or programming unit.
- Unit Testing
  - Involves verifying that each unit meets its necessary requirement or specification.

Phases of waterfall model

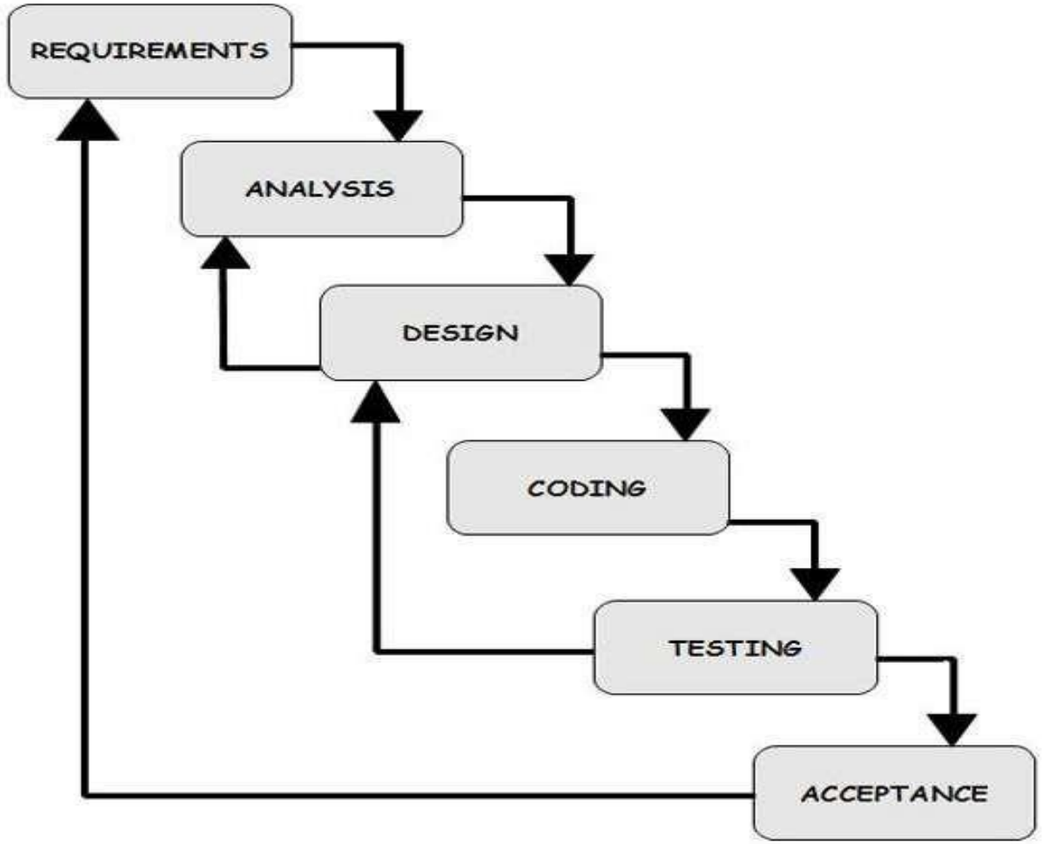
- Integration and system testing
  - The individual program units are integrated and tested as a one complete system to ensure that all the necessary requirements have been met.
  - After testing it is delivered to the customer.
- Operation and Maintenance
  - Longest life cycle phase
  - The system which has been delivered is put to the practical use
  - Involves correcting the encountered errors which were not discovered at earlier stages
  - Also by enhancing the systems services as new requirements are discovered

INTRODUCTION TO SOFTWARE ENGINEERING > WATERFALL MODEL

Traditional Model



Practitioner's Model



Advantages

- Easy to understand and Implement
- Widely used and known
- Reinforces good habits: define-before-design
- Identifies deliverables and milestones
- Document driven
- Works well on mature products and weak teams
- Fits other engineering process models

Disadvantages

- Idealized, doesn’t match reality well
- Does not reflect iterative nature of exploratory development
- Unrealistic to expect accurate requirements early on in the project
- Delays discovery of errors
- Difficult to integrate risk management
- Difficult to accommodate change after the process is underway
- One phase has to be competed before moving to the next phase.
- Difficult and expensive to make changes to document

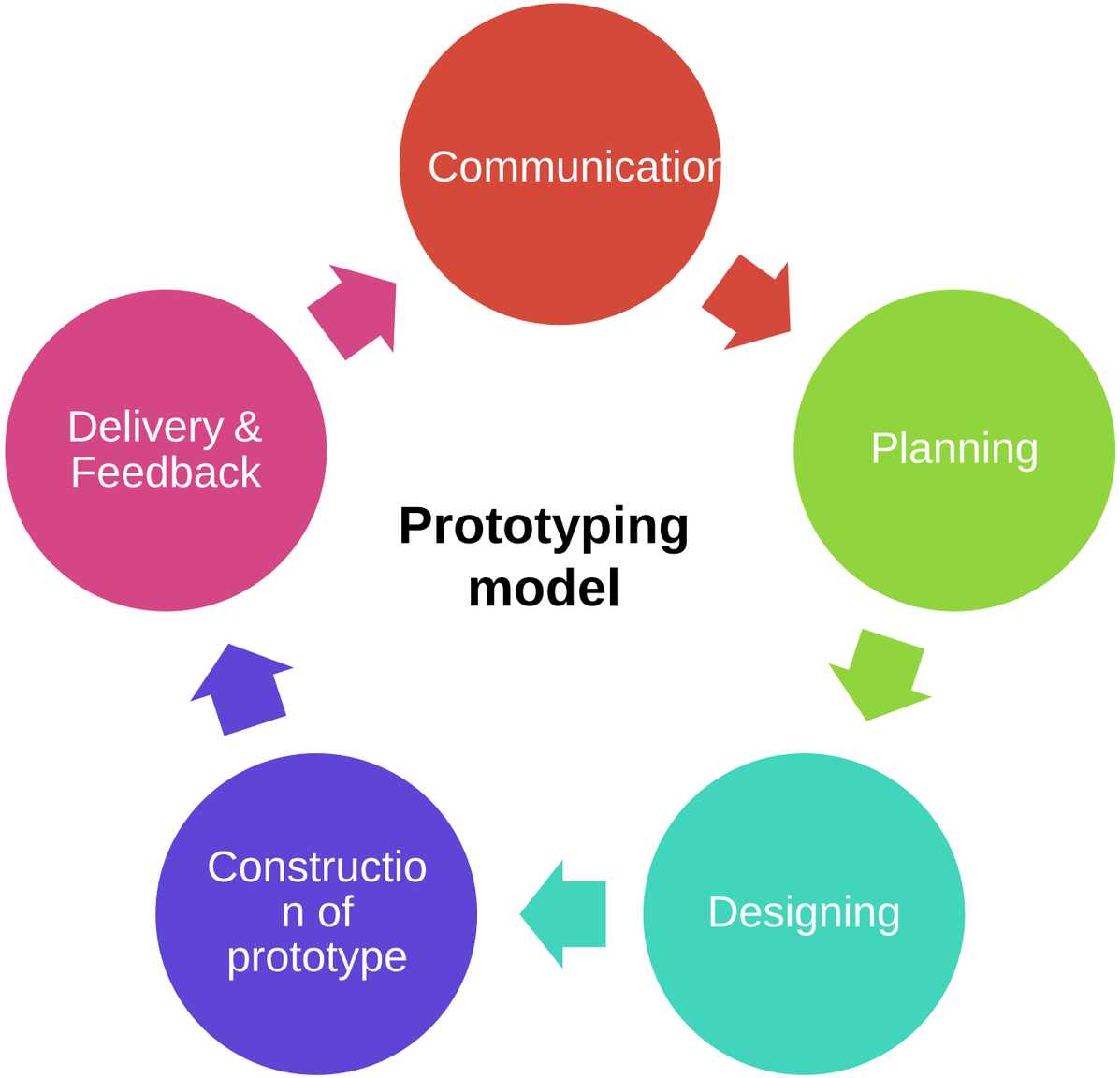
INTRODUCTION TO SOFTWARE ENGINEERING > WHEN TO USE THE WATERFALL MODEL

- Requirements are very well known
- Product definition is stable
- Technology is understood
- New version of an existing product
- Porting an existing product to a new platform.

- Prototyping model

- It is an effective paradigm for software engineering.
- Generally in this model, developers listen to the customer, design a prototype model and finally test it.
- This is repeated until all requirements of system are identified.

INTRODUCTION TO SOFTWARE ENGINEERING > PROTOTYPING MODEL



Advantages

- Users are actively involved in the development
- Errors can be detected much earlier
- Quicker user feedback is available
- Missing functionality can be identified easily
- Confusing or difficult functions can be identified

Drawbacks

- Leads to implementing and then repairing way of building systems
- Practically this methodology may increase the complexity of the system as scope of system may expand beyond original plans
- Incomplete or inadequate problem analysis

INTRODUCTION TO SOFTWARE ENGINEERING > When to use Prototype Model?

- Requirements are unstable or have to be clarified
- As the requirements clarification stage of a waterfall model
- Develop user interfaces
- Short-lived demonstrations
- New, original development

Spiral Model

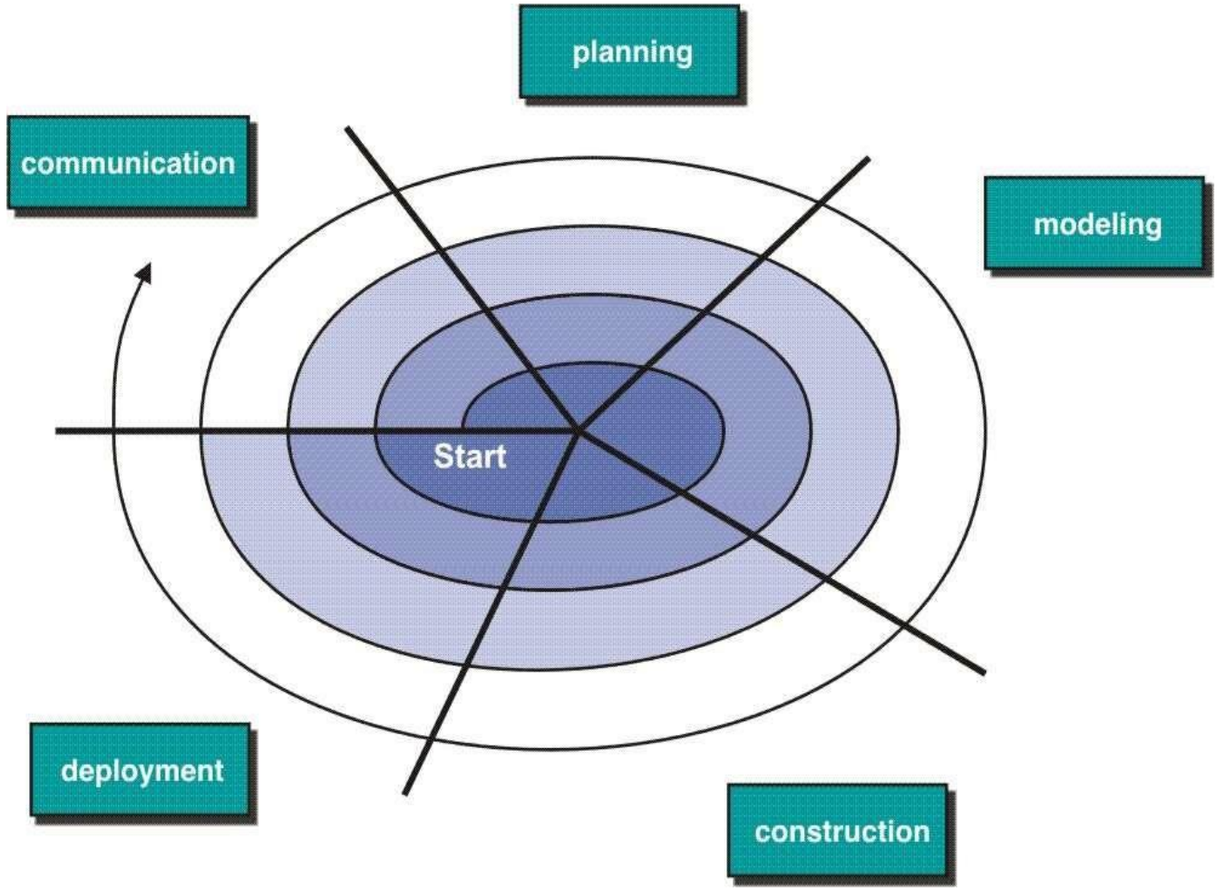
- Combines the idea of iterative development with the systematic, controlled aspects of waterfall model.
- It has four phases
  - Planning
  - Risk analysis
  - Engineering
  - Evaluation

Spiral Model - 2

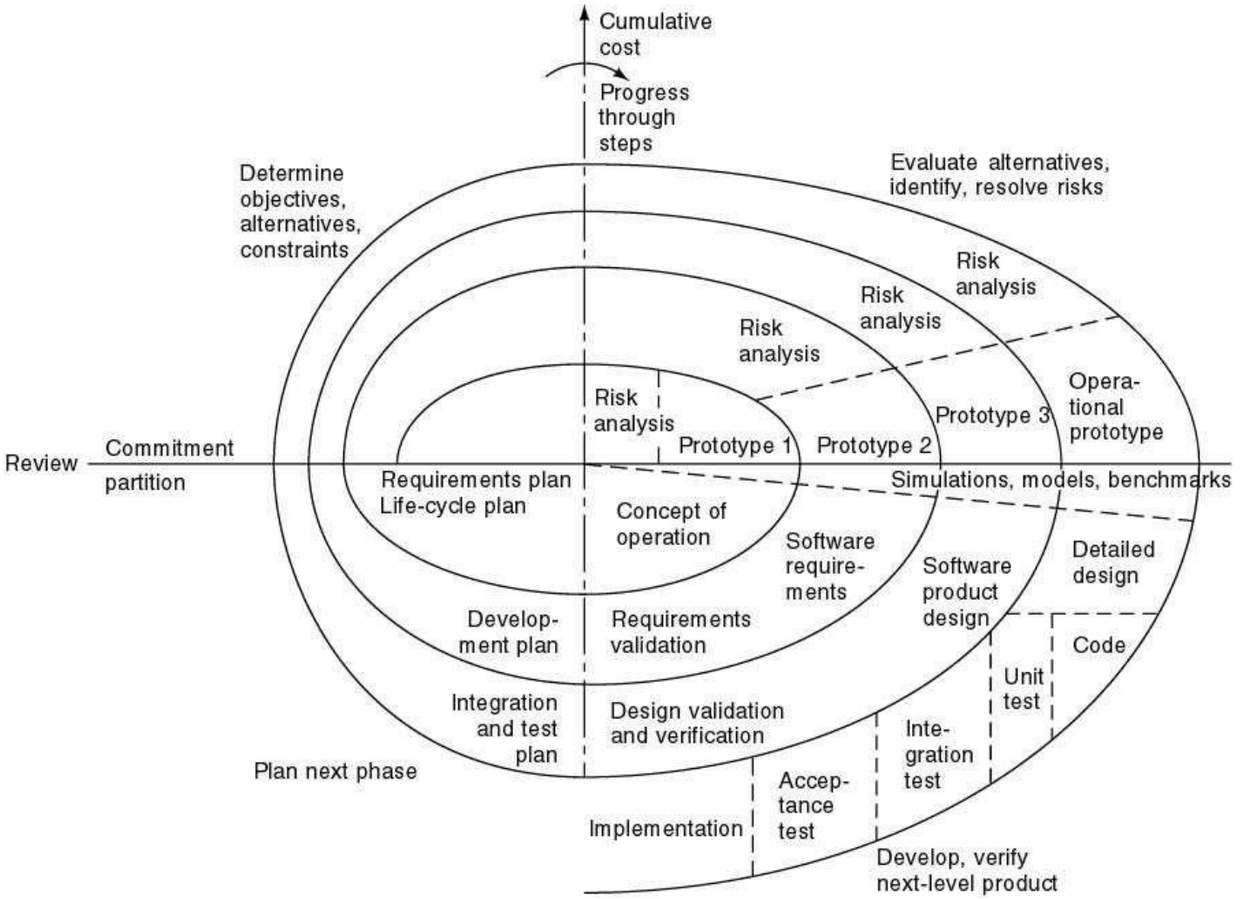
- Starting is the planning phase where the requirements are gathered and assessment of risk is done.
- In risk analysis phase the identification of risk is done and alternative solution is found.
- A prototype is produced at the end of risk analysis phase.
- Software is delivered at engineering phase along with the testing at the end of the phase that is evaluation which in turn allows the customer to evaluate his output of the project to that date.

INTRODUCTION TO SOFTWARE ENGINEERING > SPIRAL MODEL

Spiral Model



Detailed One



Advantages

- High amount of risk is resolved
- Software is produced at the early stage in the software life cycle

Disadvantages

- It may be difficult to convince customers that the evolutionary approach is controllable
- It demands considerable risk assessment
- Can be a costly model to be used
- Does not work well for small projects

INTRODUCTION TO SOFTWARE ENGINEERING > WHEN TO USE SPIRAL MODEL

- When creation of a prototype is appropriate
- When costs and risk evaluation is important
- For medium to high-risk projects
- Long-term project commitment unwise because of potential changes to economic priorities
- Users are unsure of their needs
- Requirements are complex
- New product line
- Significant changes are expected (research and exploration)

- Is driven by customer descriptions of what is required (scenarios)
- Recognizes that plans are short-lived
- Develops software iteratively with a heavy emphasis on construction activities
- Delivers frequent, multiple ‘software increments’
- Adapts as changes occur

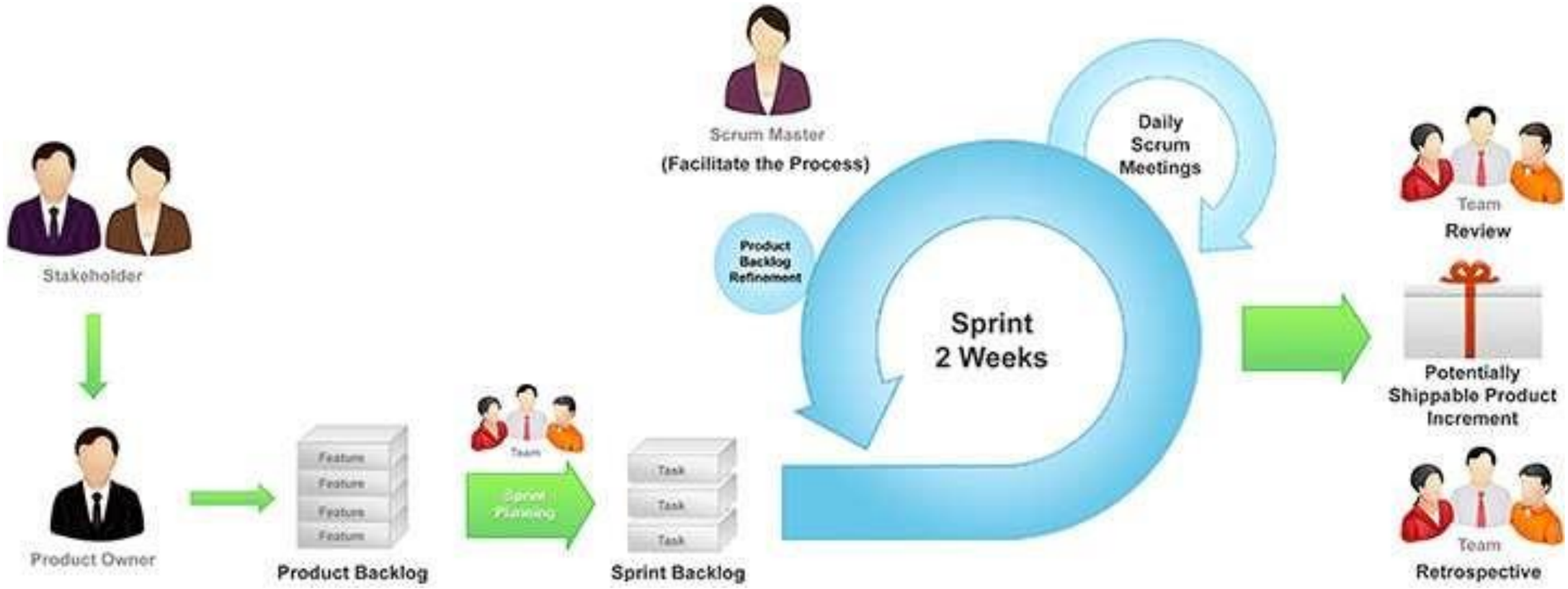
CHARACTERISTICS

- Competence
- Common focus
- Collaboration
- Decisiveness
- Fuzzy problem-solving ability
- Mutual trust and respect
- Self-organization

- Small, Highly motivated project team also called Agile Team, adopts many of the characteristics of successful software projects.
- Team members must have trust in one another.
- The distribution of skills must be appropriate to the problem.
  Unconventional person may have to be excluded from the team, if team organized is to be maintained.
- Team is “self-organizing”
- An adaptive team structure
- Uses elements of organizational paradigm’s random, open, and synchronous paradigms
- Significant autonomy

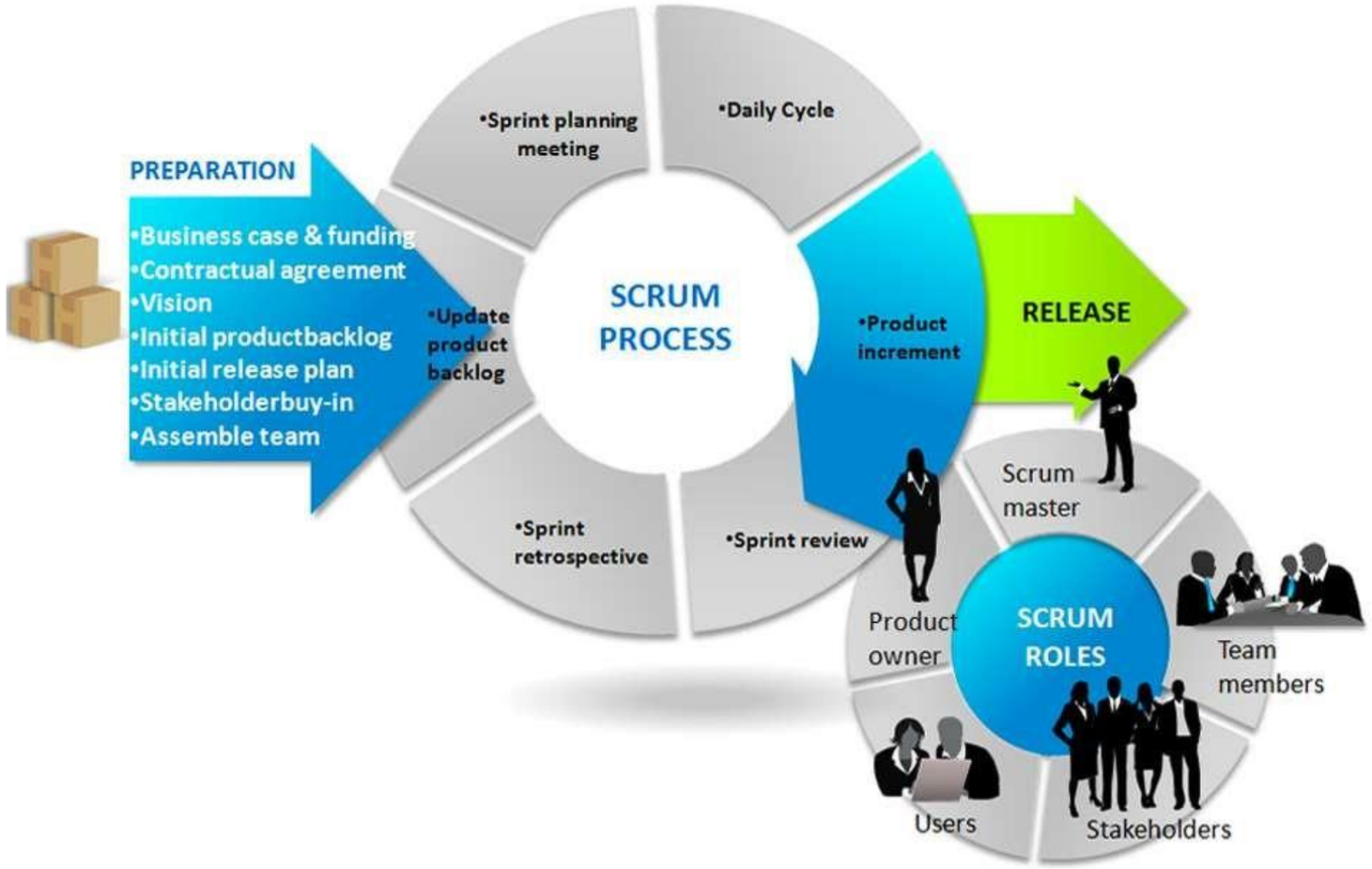
- Small working teams are organized to “maximize communication, sharing of Ideas, minimize overhead.
- The process produces frequent software increments “ that can be inspected, adjusted, tested, documented, and built on”
- Scrum—distinguishing features
  - Development work is partitioned into “packets”
  - Testing and documentation are on-going as the product is constructed
  - Scrum incorporates the following activities:
    - Requirements, Analysis, Design, Evolution, & Delivery
  - Work within activities occurs in “sprints” and is derived from a “backlog” of existing requirements

SCRUM


Stakeholder
Product Owner
Feature
Feature
Feature
Feature
Product Backlog
Team
Sprint Planning
Task
Task
Task
Sprint Backlog
Scrum Master
(Facilitate the Process)
Product Backlog Refinement
Daily Scrum Meetings
Sprint 2 Weeks
Team
Review
Potentially Shippable Product Increment
Team
Retrospective

- Backlog:
  - A prioritized list of project requirements or features that provides business value for the customer.
  - Items can be added to the backlog at any time.
  - The product manager assesses the backlog and updates priorities as required.
- Sprints:
  - Consists of work units that are required to achieve a requirement.
  - during the sprint, the backlog items that the sprint work units addresses are frozen (changes are not allowed).
  - The sprint allows team members to work in a short-term, but stable environment.

Scrum


PREPARATION
•Business case & funding
•Contractual agreement
•Vision
•Initial productbacklog
•Initial release plan
•Stakeholderbuy-in
•Assemble team
•Update product backlog
•Sprint planning meeting
•Daily Cycle
SCRUM PROCESS
•Product increment
RELEASE
•Sprint review
•Sprint retrospective
SCRUM ROLES
Scrum master
Team members
Product owner
Users
Stakeholders

- Scrum meetings
- Short meetings held daily by the scrum team.
- Three key Q's are asked & answered by all team members
- What did you do since the last team meeting?
- What obstacles are you encountering?
- What do you plan to accomplish by the next team meeting?
- A team leader called a "scrum master" leads the meeting & assess the responses from each person
- Demos
- Deliver software increment to the customer
- Customer evaluates the functionality

INTRODUCTION TO SOFTWARE ENGINEERING

GENERAL TECHNOLOGIES

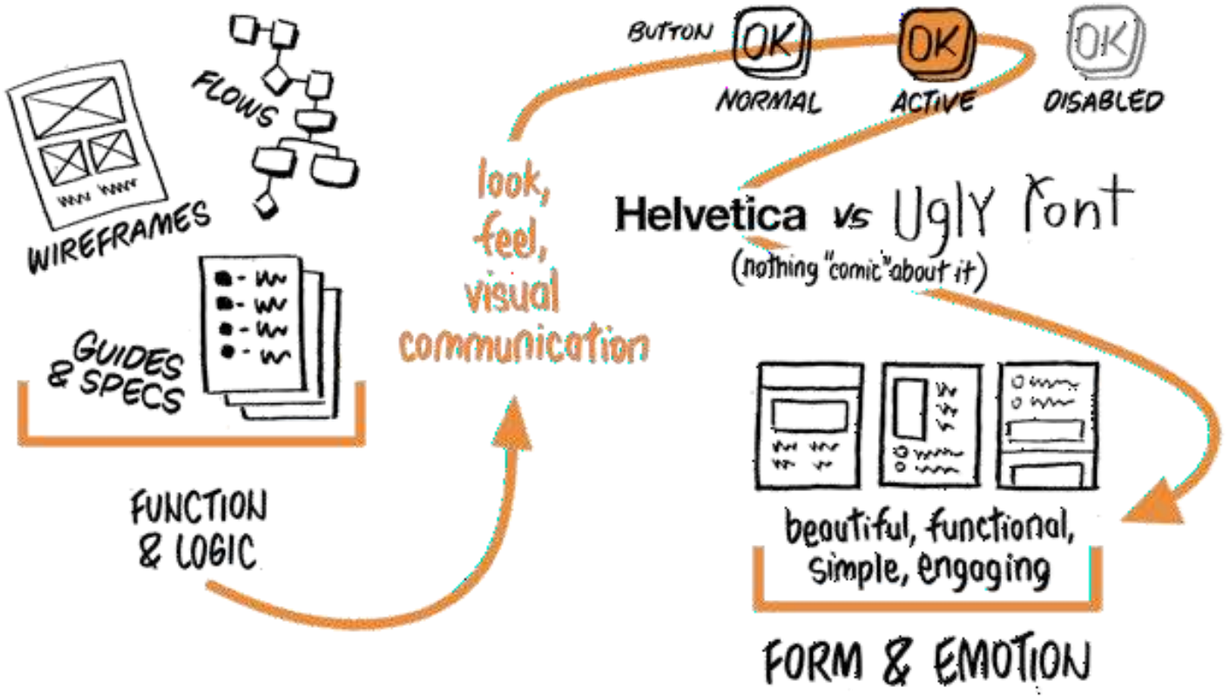
- Design Considerations
- UI
  - Enhance the UX
- Security
  - Prevent and sustain attacks
- Architecture
  - Chose a feasible model for deployment

- Effective Modular Design
- Effective modular design consist of three things:
  - Functional Independence
  - Cohesion
  - Coupling

- Design Quality Attributes
- Functionality – is assessed by evaluating the feature set and capabilities of the program.
  - Functions that are delivered and security of the overall system.
- Usability – assessed by considering human factors, consistency & documentation.
  - Reliability – evaluated by
  - measuring the frequency and severity of failure.
  - Accuracy of output results.
  - Ability to recover from failure and predictability of the program.
- Performance - measured by processing speed, response time, resource consumption, efficiency.
- Supportability – combines the ability to extend the program (extensibility), adaptability and serviceability.

UX- Design - Usability

- UX – User Experience
- The process of enhancing user satisfaction by improving the usability
- User experience design encompasses traditional human–computer interaction



UX- Design - Usability

- The developed application should fit/work in Desktop, Laptop, mobile, tabs
- Also Called Responsive Web Design (RWD)
- Twitter came with Bootstrap

Golden Rules for UI design

- Place the user in control.
  - Provide for flexible interaction.
  - Interaction should be interruptible and undoable
  - Hide technical internals from the casual user
- Reduce the user's memory load.
  - Establish meaningful defaults
  - Define shortcuts that are intuitive
- Make the interface consistent.
  - Maintain consistency across a family of applications.
  - Allow the user to put the current task into a meaningful context

Social Networking

- Applications or websites which network people who share interests, activities, backgrounds or real-life connections
- Each user will have profile
- Most social network services are web-based and provide means for users to interact over the Internet

Mobile Technology

- The technology used for cellular communication.
- Mobile technology like CDMA/GSM technology has evolved rapidly over past few years.
- Many Mobile OS are available for smartphone: Android, BlackBerry OS, webOS, iOS, Symbian, Windows Mobile Professional, etc.

Analytics

- Analytics is the discovery and communication of meaningful patterns in data.
- Especially valuable in areas rich with recorded information, analytics relies on the simultaneous application of statistics, computer programming and operations research to quantify performance.

Cloud Computing

- Cloud computing relies on sharing of resources to achieve coherence and economies of scale, similar to a utility (like the electricity grid) over a network.
- Cloud computing, or in simpler shorthand just "the cloud", also focuses on maximizing the effectiveness of the shared resources.

INTRODUCTION TO SOFTWARE ENGINEERING

CAST
(Software Analysis and Measurement Company)
Provides Application Intelligence Platform (AIP)

- An automated system for measuring the quality and size of business applications
- The AIP inspects the source code, identifies and tracks quality issues, and provides the data to monitor development performance

- **Robustness** - indication of the likelihood that an application will incur defects, corrupt data or completely fail in production. Robustness Health Factor as an index from 1 to 4, with 4 indicating the highest level of stability.
- **Functional size** - measure of the amount of business function within applications, expressed as function points
- **Efficiency** - measure of potential performance and scalability bottlenecks in software

- **Complexity** - *CAST measures software complexity by evaluating the level of cyclomatic complexity, essential complexity, SQL complexity, coupling and integration complexity using system-level static analysis*
- **Technical debt** - *measures the accumulated amount of rework that is needed to correct or recover from mistakes made and short cuts taken during the development process*

- **Risk** - CAST measures risk by evaluating the violations of industry-based best practices within the code, components, and architecture of applications.
- **Changeability** - measures how flexible and adaptable the application is when it is getting enhanced
- **Application portfolio health** - With CAST HIGHLIGHT you can inventory 100+ applications in a week, and monitor your portfolio over time to trend application size, risk, complexity, technical debt and software maintenance. It is a cloud-based solution.

Key points discussed in this course:

- Software Engineering and its importance
- Software process and different types of models
- Recent general technological developments