Software Engineering

Amrita University

What is Software Engineering?

- Branch of computer science that seeks principles to guide the development of large, complex software systems.
- Problems faced are more when compared to problems faced when writing small programs.
- Development of large systems requires the effort of more than one person over an extended period of time.
- Requirements may be altered, personnel assigned may change during the period.

What does SE contain?

- Includes topics such as
  - Personnel management and
  - project management
- The above topics are more readily associated with business management than computer science.
- We will focus on topics related to computer science

The Software Engineering Discipline

- Consider a large complex device such as
  - an automobile,
  - a multi-story office building, or
  - perhaps a cathedral

- What are the questions to be answered during the development of such a large software system?

Questions to be answered

- How can you estimate the cost in time, money, and other resources to complete the project?
- How can you divide the project into manageable pieces?
- How can you ensure that the pieces produced are compatible?
- How can those working on the various pieces communicate?
- How can you measure progress?
- How can you cope with the wide range of detail (the selection of the doorknobs, the design of the gargoyles, the availability of blue glass for the stained glass windows, the strength of the pillars, the design of the duct work for the heating system)?

Advancing the software engineering discipline

- Engineering is well-established.
- Wealth of previously developed engineering techniques may exist.
- May be useful in answering the above questions.
- Still s/w engineering products are challenged leading to cost overrun, late delivery of products and dissatisfied customers

Distinction between SE and other fields

- Lack of ability to construct systems from generic prefabricated components like traditional fields of engineering (E.g., automobile engineering)
- Lack of quantitative techniques, called **metrics,** for measuring the properties of software
- Methods for measuring the “complexity” of software are evasive.
- Evaluating the quality of a software product is challenging (software doesn’t wear out)

Research in software engineering

- 2 Levels
- developing techniques for immediate application – by practitioners
  - Subjective foundation – Methodologies obsolete
- search for underlying principles and theories on which more stable techniques can someday be constructed – theoreticians
  - Progress continues to be slow

Societal Addiction to s/w systems

- Economy, healthcare, government, law enforcement, transportation and defence depend on large software systems
- Reliability – Major problem
- S/w errors cause disasters and near disasters
  - Rising moon interpreted as nuclear attack
  - one-day loss of $5 million by the Bank of New York
  - the loss of space probes
  - radiation overdoses that have killed and paralyzed
  - Simultaneous disruption of telephone communications over large regions

Computer-aided software engineering (CASE)

- Application of computer technology to the software development process
- Streamline and otherwise simplify the software development process
- Has led to the development of a variety of computerized systems, known as **CASE tools**

CASE Tools

- Include
  - project planning systems (to assistin cost estimation, project scheduling, and personnel allocation),
  - project management systems (to assist in monitoring the progress of the development project),
  - documentation tools (to assist in writing and organizing documentation),
  - Prototyping and simulation systems (to assist in the development of prototypes),
  - Interface design systems (to assist in the development of GUIs),
  - programming systems (to assist in writing and debugging programs)

Integrated development environments (IDEs)

- Combine tools for developing software (editors, compilers, debugging tools, and so on) into a single, integrated package.
  - examples are
    - systems for developing smartphone applications.
      - Not only provide the programming tools necessary to write and debug the software
      - But also provide simulators
        - » by means of graphical displays allow a programme to see how the software being developed would actually perform on a phone.

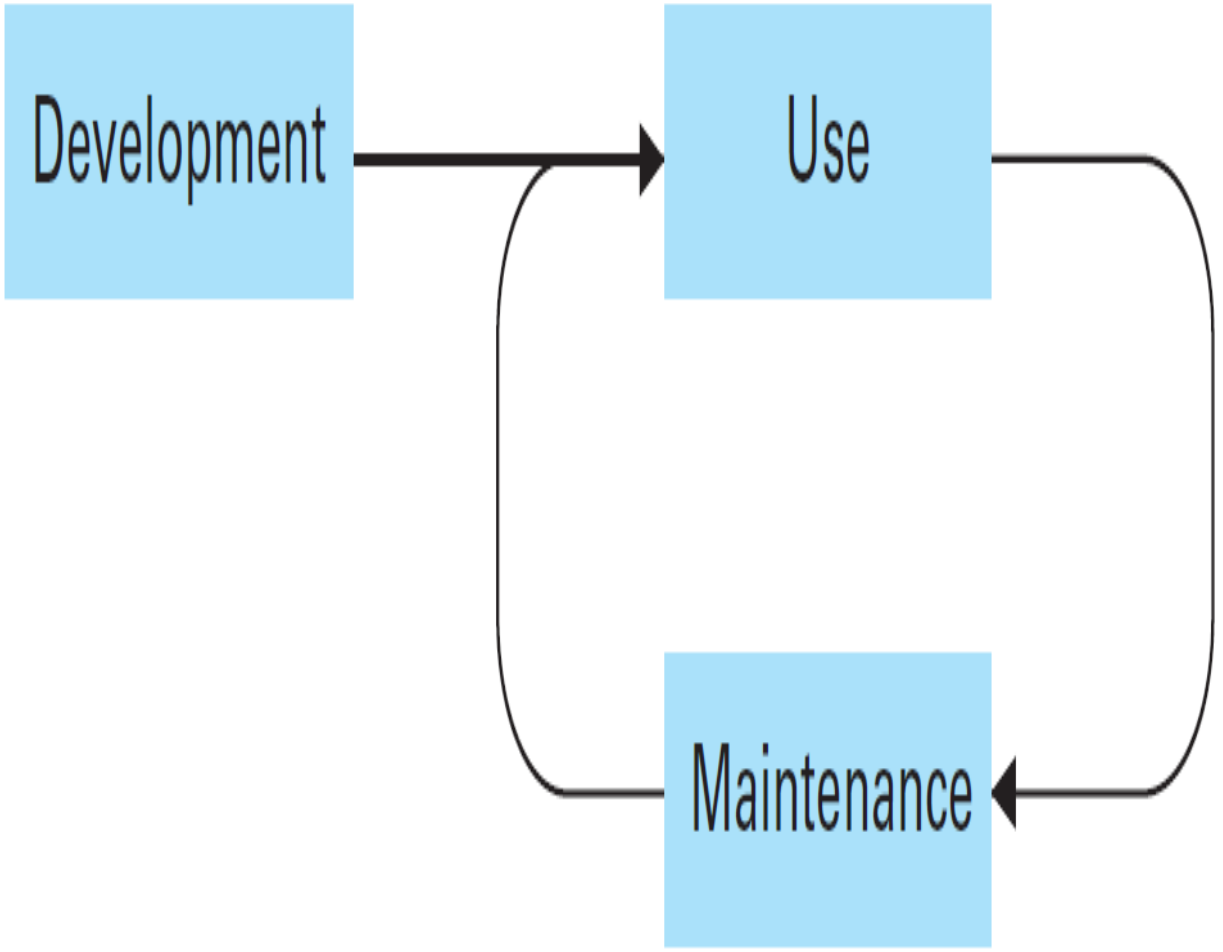
Improving the state of software engineering

- In addition to researchers
- ISO (International Standards Organization)
- ACM (Association for Computing Machinery)
- IEEE (Institute of Electrical and Electronics Engineers)

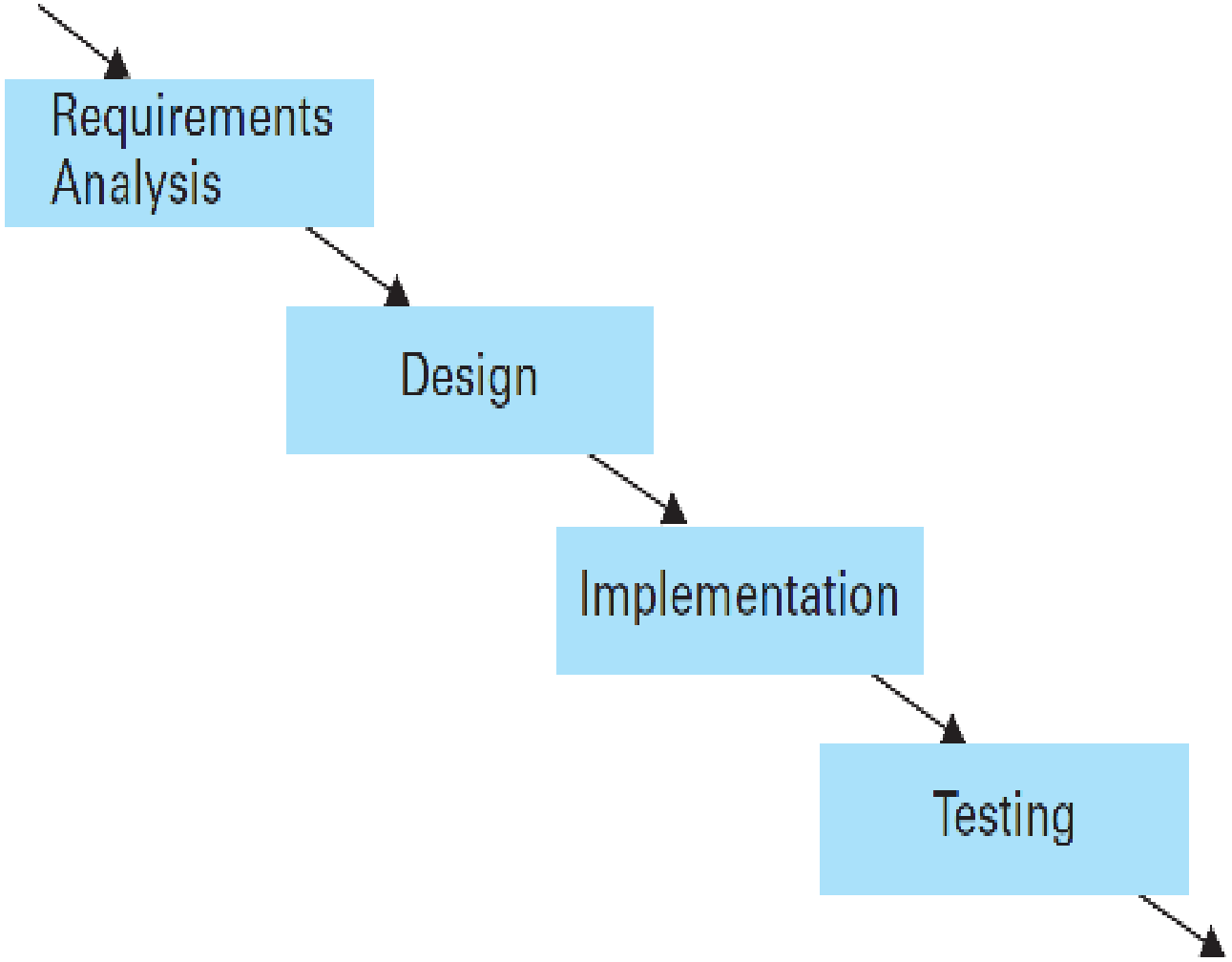
Efforts by Organizations

- Range from
  - adopting codes of professional conduct and ethics
  - enhance the professionalism of software developers
  - counter nonchalant attitudes toward each individual's responsibilities
  - establishing standards for measuring the quality of software development organizations and
  - providing guidelines to help these organizations to improve their standings.

The Software Life Cycle



Traditional Development Phase



Goals of Requirements Analysis

- To specify what services the proposed system will provide.
- To identify any conditions (time constraints, security, and so on) on those services.
- To define how the outside world will interact with the system.

Requirements Analysis Process

- Input from the **stakeholders of the system**
  - If the user is a company or govt. agency, a software developer may be hired for the execution of the project
    - Process starts with the feasibility study conducted by the user
  - If the developer is producing commercial off-the-shelf (COTS) software for the mass market, (sold in retail stores or downloaded via the Internet)
    - Process starts with a market study by the developer

Requirements Analysis Process

- Compiling and analysing the needs of the software user
- Negotiating with the project's stakeholders over trade-offs between
  - wants,
  - needs,
  - costs and
  - feasibility
- Finally developing a set of requirements that identify the features and services that the finished software system must have.

Software Requirements Specification

- Document to record the requirements
- Written agreement between all parties concerned
- Intended to
  - guide the software's development and
  - provide a means of resolving disputes arising later

- IEEE and U.S DOD have adopted standards for the document

Qualities of SRS

- Theoretically
  - Should define a firm objective toward which the software's development can proceed
- In Practice
  - fails to provide this stability
  - poor communication and changing requirements are the major causes of
    - cost overruns
    - late product delivery

Design

- Creating a plan for the construction of the proposed system

Requirements Analysis	Design
Identifying the problem to be solved	Is about developing a solution to the problem
What a software system is to do	How the system will do it

Flawed Design

- Lot of *how* is considered during requirements analysis
- Lot of *what* is considered during design

Design

- Internal structure of software is established
- Result is a detailed description of the system structure such that
- Design can be converted into programs
- Example: Office building design
  - Detailed structural plan meeting specific requirements
  - Blueprints at various levels of detail
  - Standard notations and model diagrams are used
  - In S/w notational systems are not stable
  - Very dynamic and researchers struggle to find better approaches

Implementation

- Actual writing of programs
- Distinction between software analyst and programmer
- Software analyst: Involved in entire development process with emphasis on requirements analysis and design
- Programmer writes programs and implements the design produced by s/w analyst
- Many programmers are analysts and many analysts are programmers

Testing

- Traditionally in the past – Debugging of programs and
- Confirming whether the final s/w product is compatible with SRS
- Today each step in the entire development process is tested
- Only step for quality assurance in the entire s/w development life cycle
- Testing hence should not be a separate step but must be incorporated into requirements analysis and confirmation, design and validation and implementation and testing

Goals of software engineering

- Even with modern quality assurance
  - Large s/w systems contain errors
  - Even after significant testing
  - Many errors go undetected for the entire life of the system
  - Other errors cause major malfunctions
  - Elimination of such errors is the goal of s/w engg
  - The prevelance of error indicates that a lot of research has to be done here

Software Engineering Methodologies

- Water fall model
- Incremental model
- Iterative model
- Agile Methods

Waterfall model

- Entire Requirement specification must be completed before design
- Entire design must be completed before implementation and so on.
  - Process flows only in one direction
  - Allowing too much variations was a risk in a large software system was the belief
  - So all the processes were done in a strictly sequential manner

Incremental model

- Free wheeling trial and error process of problem solving
- Software is constructed in increments
  - a simplified version of the final product with limited functionality
  - Once tested and evaluated by the user, more features are added and tested until the entire system is complete
  - Example patient record system for a hospital with view feature for a sample records initially and later on add and update records are handled in an incremental manner

Iterative model

- Similar to incremental model
- Incremental model extends the previous version
- Iterative model refines each version
- Incremental model includes iteration and iterative model may incrementally add features

Iterative Techniques

- Rational Unified Process (RUP)
  - Created by the Rational Software Corporation
  - Now a division of IBM
- A software development paradigm
- Redefines the steps in the development phase of the software life cycle
- Provides guidelines for performing those steps
- Guidelines, along with CASE tools to support them, are marketed by IBM

Unified Process

- RUP is widely applied throughout the software industry
- Unified process
  - Developed by the popularity of RUP
  - A non-proprietary version of RUP
  - Available on a non-commercial basis

Prototyping

- Incomplete versions of the proposed system are built and evaluated
- Used by incremental and iterative models
- Incremental models
  - Prototypes evolve into the complete, final system
  - Process known as **evolutionary prototyping**
- Iterative Situations
  - Prototypes may be discarded in favour of a fresh implementation of the final design
  - Approach is known as **throwaway prototyping**

Rapid Prototyping

- Falls into throwaway category
- A simple example of the proposed system is quickly constructed in the early stages of development
  - Example: a few screen images
  - Gives an indication of how the system will interact with its users and
  - what capabilities the system will have

  Goal: not to produce a working version of the product

Rapid Prototyping Goals

- Not to produce a working version of the product
- To obtain a demonstration tool that can be used to clarify communication between the parties involved in the software development process
  - Have proved advantageous in clarifying system requirements during requirements analysis or
  - As aids during sales presentations to potential clients.

Open-source development

- A less formal incarnation of incremental and iterative ideas
- Used for years by computer enthusiasts / hobbyists
- Example: Linux Operating system
  - Open-source development was originally led by Linus Torvalds

Open-source development

- A single author writes an initial version of the software (for his/her needs)
  - posts the source code and its documentation on the Internet
  - Downloaded and used by others without charge
  - Other users modify or enhance the software to fit their own needs or to correct errors that they find
  - Changes are reported to original author
  - Incorporated into the posted s/w version
  - Extended version available for further modifications
  - s/w package evolves thru several extensions in single week

Agile Methods

- Pronounced shift from the waterfall model
- Represented by a collection of methodologies each of which proposes
  - early and quick implementation on an incremental basis,
  - Responsiveness to changing requirements, and
  - a reduced emphasis on rigorous requirements analysis and design.
  - Example: Extreme Programming (XP)

Extreme Programming (XP)

- Software is developed by a team of less than a dozen individuals
  - In a communal work space
  - Freely share ideas
  - Assist each other in project development
- Software developed
  - Incrementally
  - repeated daily cycles of
    - informal requirements analysis,
    - designing,
    - implementing, and
    - testing

Extreme Programming (XP)

- New expanded versions of the software package appear on a regular basis
- each version can be evaluated by the project's stakeholders and
  - used to point toward further increments
- Agile methods are characterized by flexibility
- Stark contrast to the waterfall model where
  - managers and programmers working in individual offices while
  - rigidly performing well-defined portions of the overall software development task.

Agile Vs Waterfall Method

Agile Model	Waterfall Model
•Agile method proposes incremental and iterative approach to software design	•Development of the software flows sequentially from start point to end point.
•The **agile process** is broken into individual models that designers work on	•The design process is not broken into an individual models
•The customer has early and frequent opportunities to look at the product and make decision and changes to the project	•The customer can only see the product at the end of the project

Agile Vs Waterfall Method

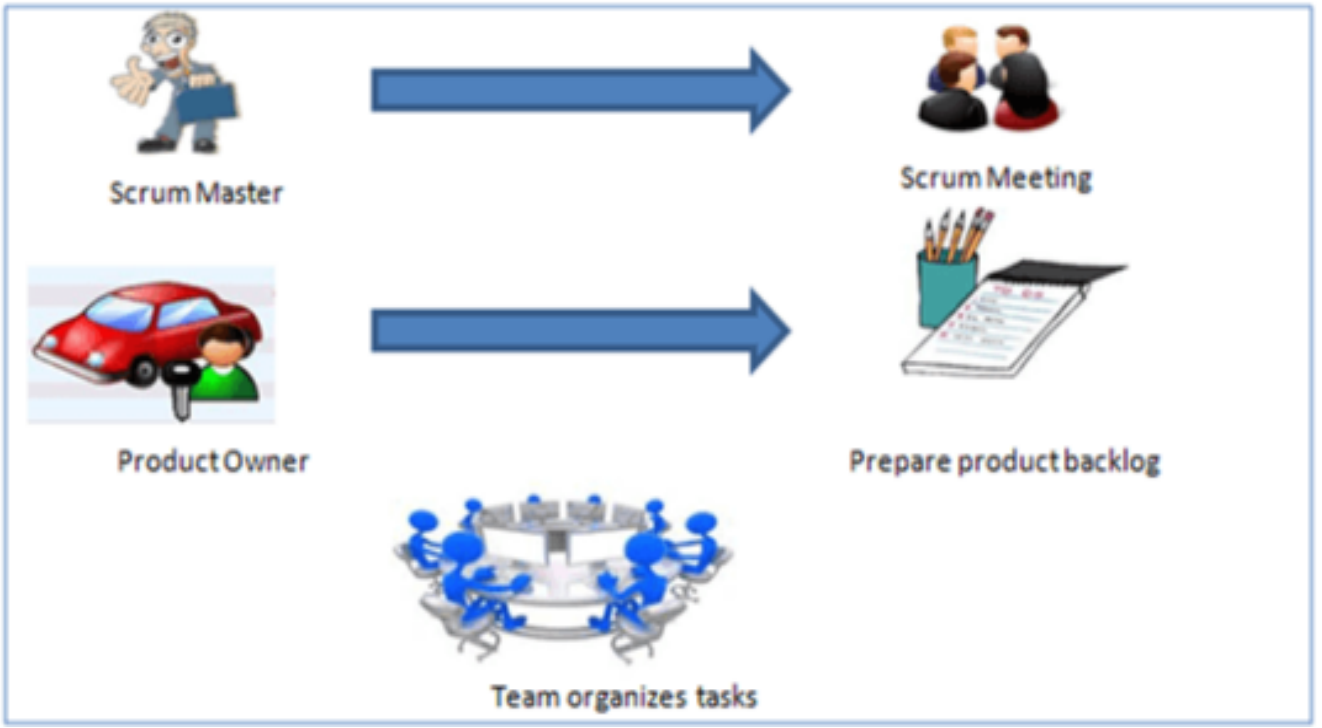
•Agile model is considered unstructured compared to the waterfall model	•Waterfall model are more secure because they are so plan oriented
•Small projects can be implemented very quickly. For large projects, it is difficult to estimate the development time.	•All sorts of project can be estimated and completed.
•Error can be fixed in the middle of the project.	•Only at the end, the whole product is tested. If the requirement error is found or any changes have to be made, the project has to start from the beginning
•Development process is iterative, and the project is executed in short (2-4) weeks iterations. Planning is very less.	•The development process is phased, and the phase is much bigger than iteration. Every phase ends with the detailed description of the next phase.

Agile Vs Waterfall Method

•Documentation attends less priority than software development	•Documentation is a top priority and can even use for training staff and upgrade the software with another team
•Every iteration has its own testing phase. It allows implementing regression testing every time new functions or logic are released.	•Only after the development phase, the testing phase is executed because separate parts are not fully functional.
•In agile testing when an iteration end, shippable features of the product is delivered to the customer. New features are usable right after shipment. It is useful when you have good contact with customers.	•All features developed are delivered at once after the long implementation phase.
•Testers and developers work together	•Testers work separately from developers
•At the end of every sprint, user acceptance is performed	•User acceptance is **performed** at the end of the project.
•It requires close communication with developers and together analyze requirements and planning	•Developer does not involve in requirement and planning process. Usually, time delays between tests and coding

Various **methods** present in agile testing

- **Scrum**
  - an agile development method which concentrates specifically on how to manage tasks within a team-based (say- 7 to 9 members) de



Roles in Scrum

- Scrum Master
  - Master is responsible for setting up the team, sprint meeting and removes obstacles to progress
- Product owner
  - The Product Owner creates product backlog, prioritizes the backlog and is responsible for the delivery of the functionality at each iteration
- Scrum Team
  - Team manages its own work and organizes the work to complete the sprint or cycle

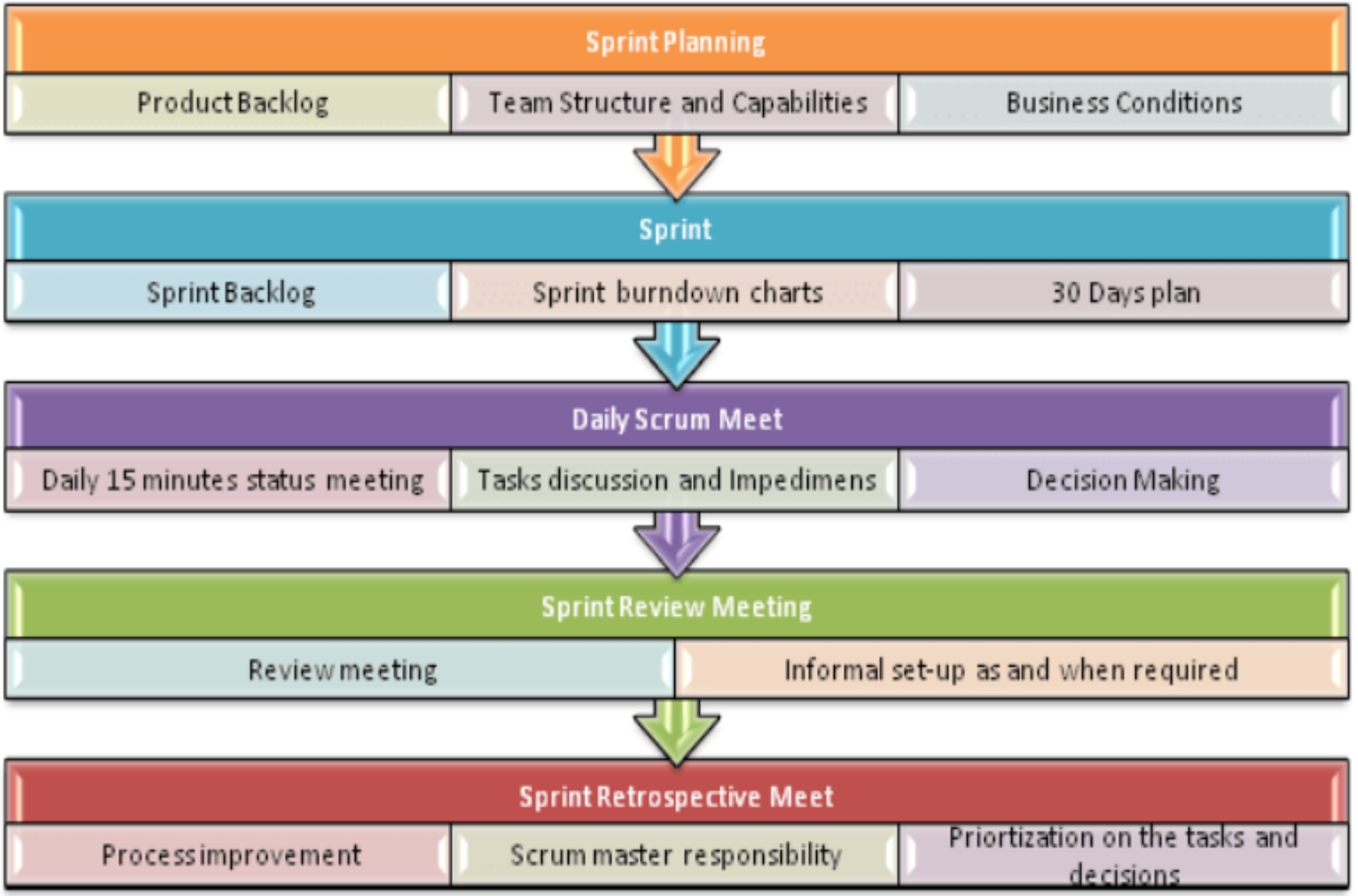
Product Backlog

- Repository where requirements are tracked with details on the no of requirements to be completed for each release.
- Should be maintained and prioritized by Product Owner, and it should be distributed to the scrum team.
- Team can also request for a new requirement addition or modification or deletion

Process flow of Scrum Methodologies

- Each iteration of a scrum is known as Sprint
- Product backlog is a list where all details are entered to get end product
- During each Sprint, top items of Product backlog are selected and turned into Sprint backlog
- Team works on the defined sprint backlog
- Team checks for the daily work
- At the end of the sprint, team delivers product functionality

Scrum Practices



Reference

- https://www.guru99.com/agile-scrum-extreme-testing.html