JAVA FULL STACK DEVELOPMENT PROGRAM

SOFTWARE DEVELOPMENT LIFECYCLE

OUTLINE

- Daily Routine of SDE
  - Software Development Life Cycle - SDLC
  - Model - Waterfall
  - Framework – Scrum
  - Development tool - Jira
  - Development Environment

SDLC

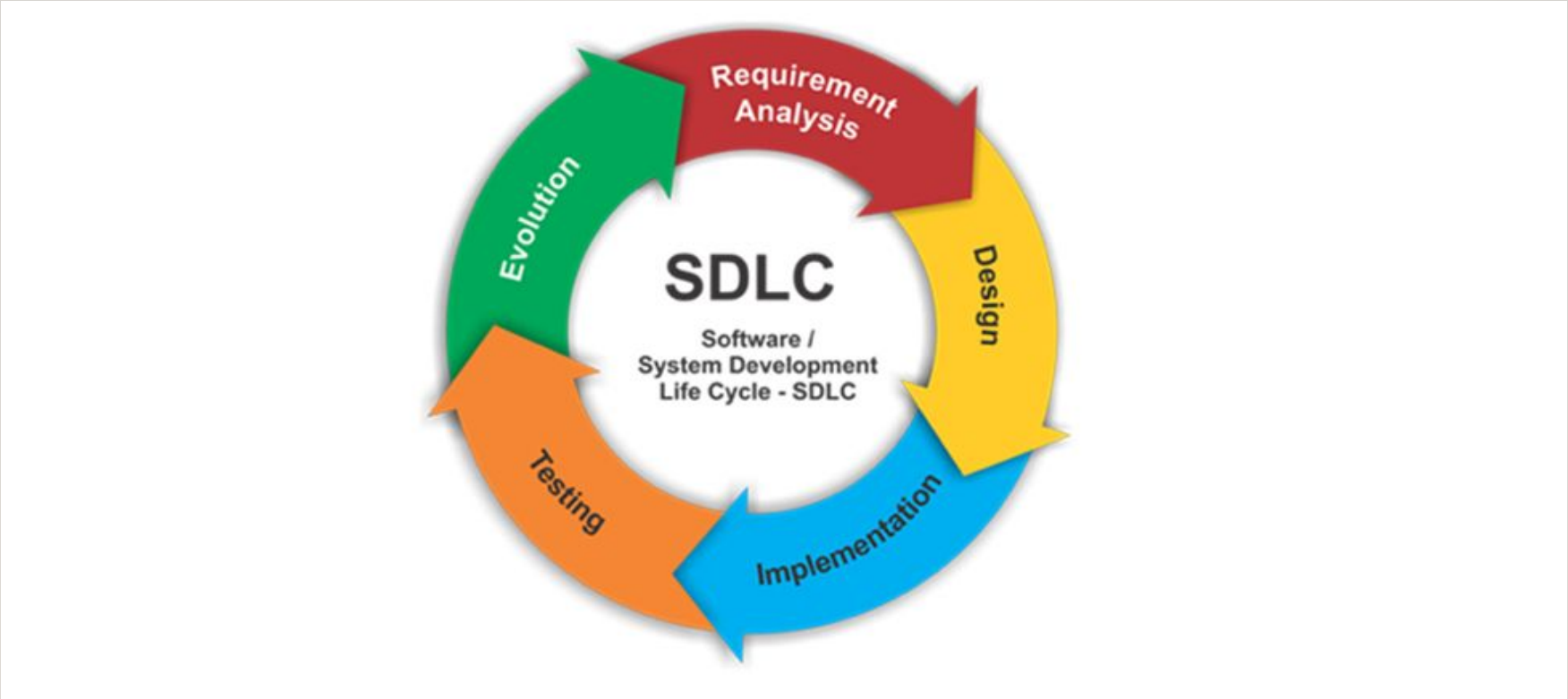
WHAT IS SDLC

- *SDLC* stands for Software Development Life Cycle
- Also known as application development life cycle
- Systematic process for planning, creating, testing and deploying

WHY DO WE NEED SDLC

- Produce high-quality software that meets or exceed customer expectations
- Reaches completion within time and cost estimates
- With appropriate SDLC model, developers are able to:
  - Give correct estimate on time & cost
  - Organize efficient communication both inside team or between teams
  - Increase work transparency, reduce chaos
  - Help all the members better evaluate their performance

SDLC - SAMPLE


Requirement Analysis
Design
Implementation
Testing
Evolution
SDLC
Software / System Development Life Cycle - SDLC

SDLC – BASIC STEPS

- Requirement Analysis
  - What is the purpose of this application?
  - What are the functionalities it should have?
  - Is it possible to reduce the workload by reuse existing application/service?
- Design
  - Macro-level
    - What programming language should we use?
    - Is our application stand alone? Will it interact with other applications?
    - If not stand alone, is there any standard we must follow?
    - What type of framework should we choose?

SDLC – BASIC STEPS

- Design
  - Micro-level
    - How many modules should we have? What are their responsibility?
    - How should we separate the codes and functions into different logic layers?
    - What coding/testing standard should we follow?
    - What are the design patterns the entire team must follow?
- Implementation
  - Manage dependencies
  - Writing code
  - Implement unit test/integration test

SDLC – BASIC STEPS

- Testing
  - Perform local testing
  - Deploying to testing environment
  - Find/fix bugs
  - Deploy to production environment
- Evolution
  - Review newly launched features; does it work good?
  - Is there anything that can be improved?

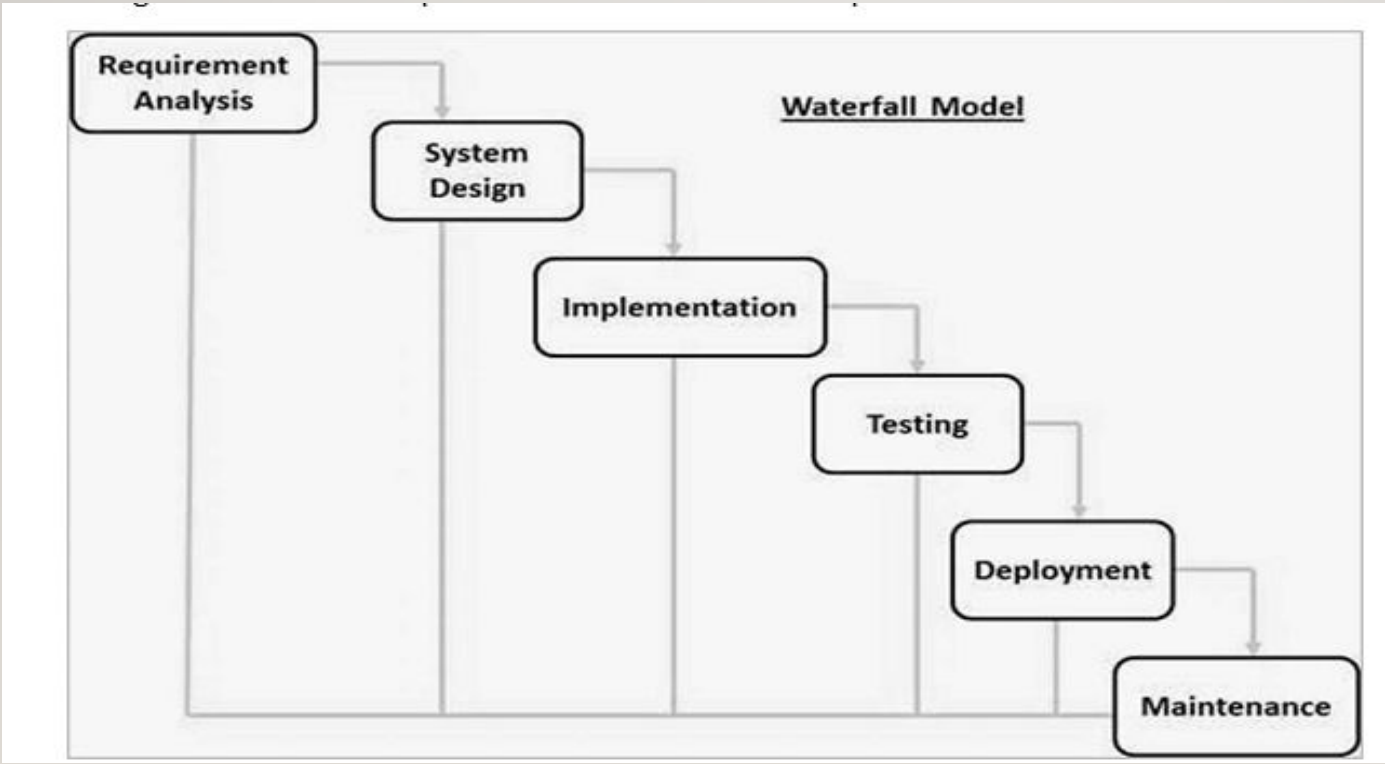
SDLC – MODELS

- SDLC Models
  - Waterfall
  - Iterative
  - Spiral
  - V-Model
  - Big-Bang
  - Agile

WATERFALL MODEL

- It is the earliest SDLC approach.
- *Each phase must be completed entirely before starting the next phase*
- When to use?
  - Requirements are well documented
  - Product definition is stable, technology is not dynamic
- Pros
  - Clear definition, simple and easy to understand
- Cons
  - Vulnerable to risk and uncertainty, hard to measure progress

WATERFALL MODEL DIAGRAM



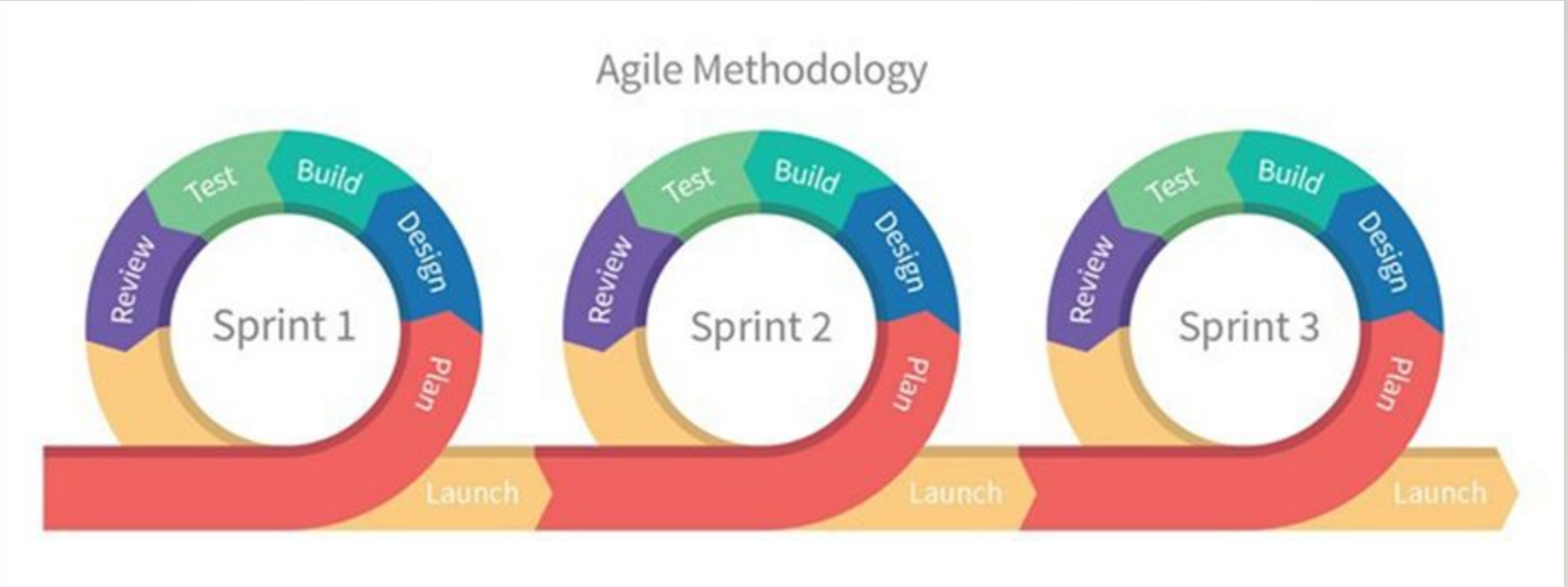
AGILE MODEL

- A SDLC model focuses on customer satisfaction with *rapid delivery* as its core idea.
- Breaks product into small incremental pieces/builds, divide and conquer in different iterations
- Each iteration will last for 2 to 4 weeks, team will focus on pre-scheduled task and deliver the result at the end of the time period

AGILE MODEL

- Pros
  - Rapid development life cycle, require less resource
  - Fast delivery, easy to manage
  - Able to give accurate estimate on timeline
- Cons
  - No suitable for complex features
  - More efforts needed in maintenance
  - Leadership is required
  - Hard to transfer project knowledge as documentation is lacking

AGILE MODEL DIAGRAM



SCRUM

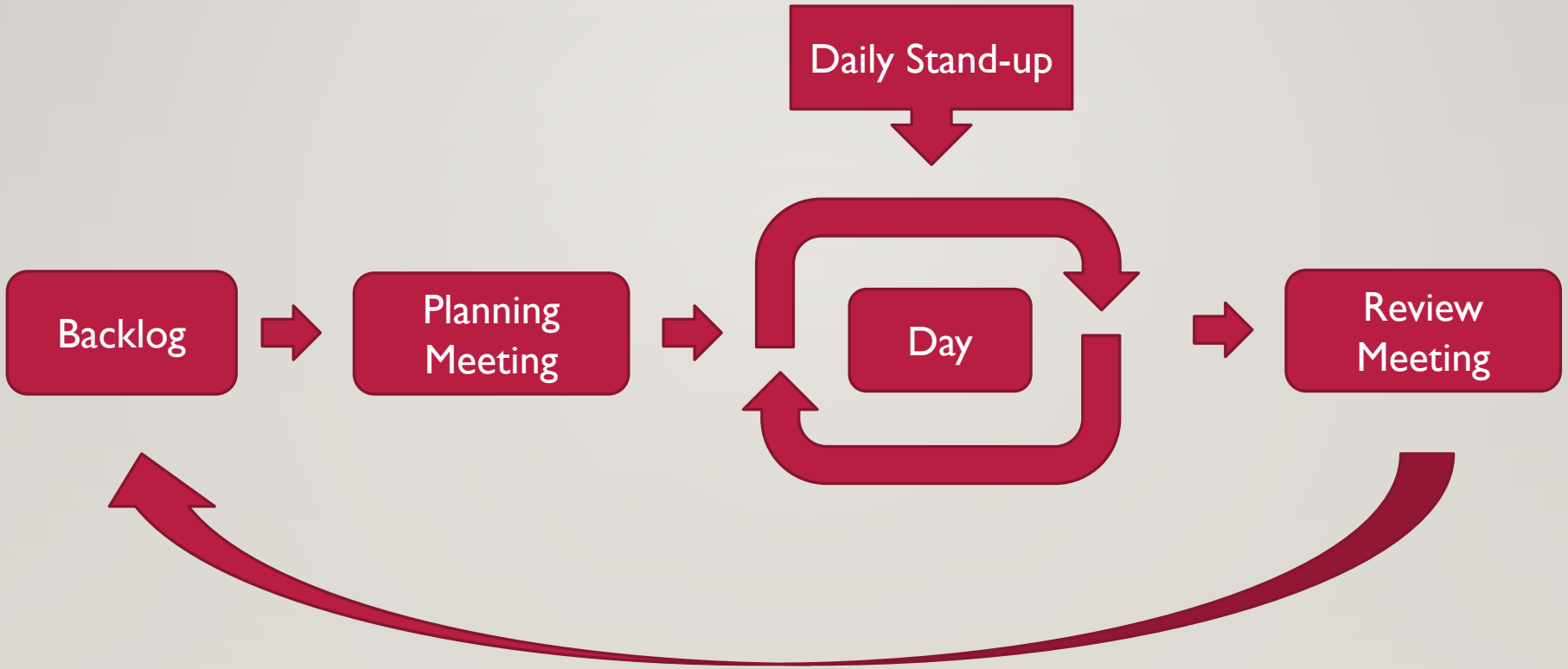
SDLC FRAMEWORK

- SDLC is principles, not actual implementation
- Need a practice or framework to implement the idea
- Java Web development – spring framework / servlet
- Agile uses SCRUM

SDLC SCRUM

- A SDLC framework used by Agile principle, popular among enterprise level development team
- **Backlog**: place where we store all the information about our application/project, including tasks that need to be accomplished.
- **Sprint**: repeated fixed time-box. Development life cycle is composed of Sprints. Each sprint tasks will be assigned to team members.
- **Meeting**: in each sprint, the team will hold multiple meetings to plan for the tasks, solve issues, and improve overall performance

SCRUM WORKFLOW



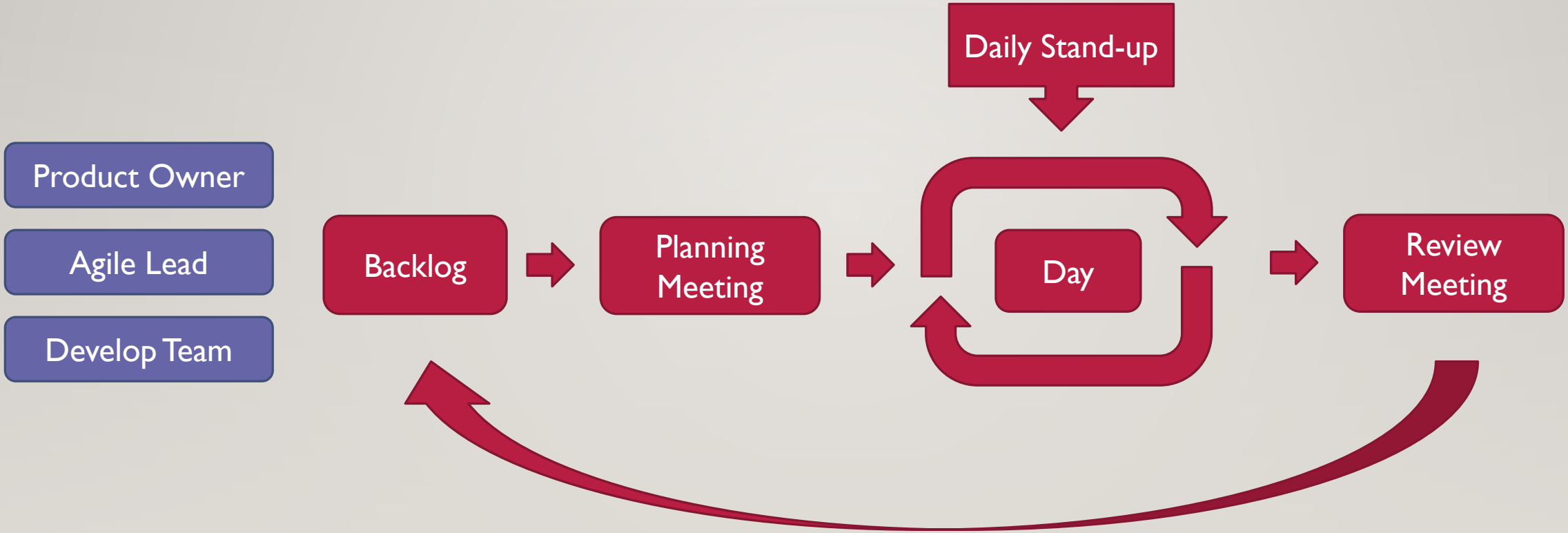
SCRUM TERMINOLOGY

- Important concepts in Scrum:
  - Three Roles
  - Three Work Pieces
  - Five Events

THREE ROLES

- Normally, an application will be assigned to a team. This team is in charge of design, maintain and make improvement.
- To apply Scrum framework, a team must have the following part:
  - ***Product Owner***: ***owner*** of the product (application). Act as the bridge between tech team and business team. Help tech team understand the business requirement and help business team understand the tech efforts needed.
  - ***Scrum Master***: also known as ***Agile lead***. Responsible for organizing meetings, checking team status and making sure everything is on schedule. Work closely with PO.
  - ***Develop team***: regular tech team. Usually consists of BE, FE and QE developers.

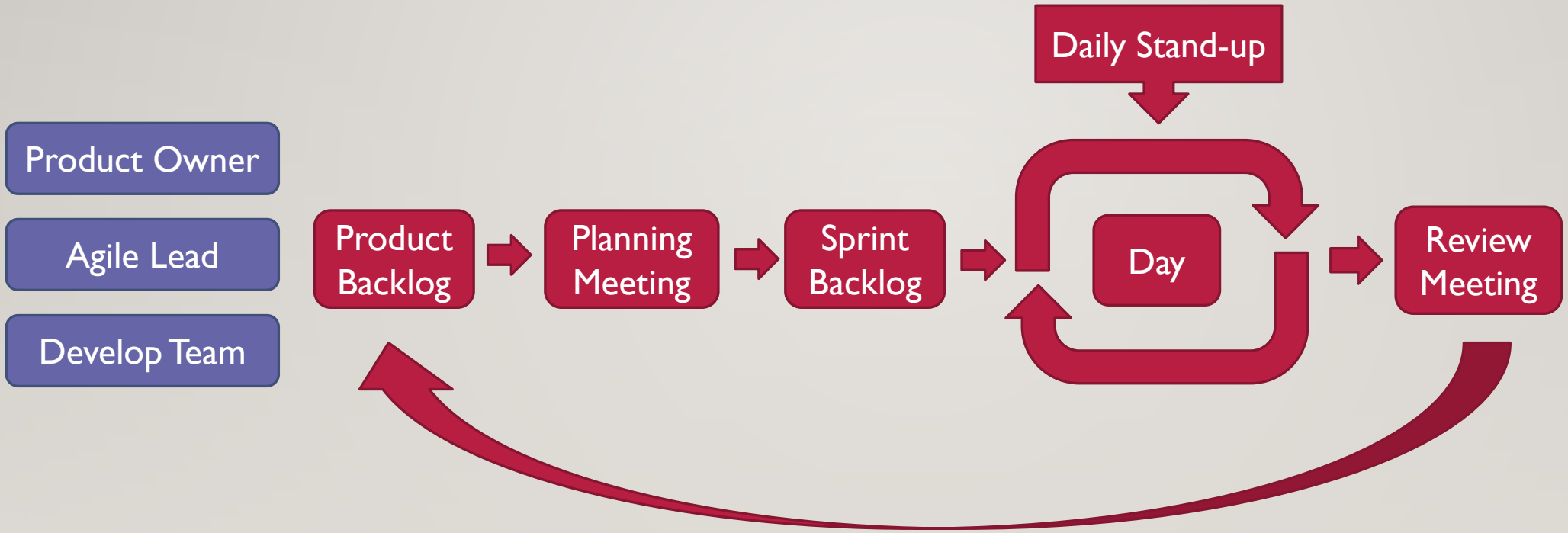
SCRUM WORKFLOW



THREE WORK PIECES

- ***Product Backlog***: contains all the tasks of this application
- ***Sprint Backlog***: subset of Product backlog. Each Sprint, PO will pick tasks from product backlog and move into sprint backlog. So the team can narrow down the range and focus on certain tasks.
- ***Increment***: list possible improvements for existing features

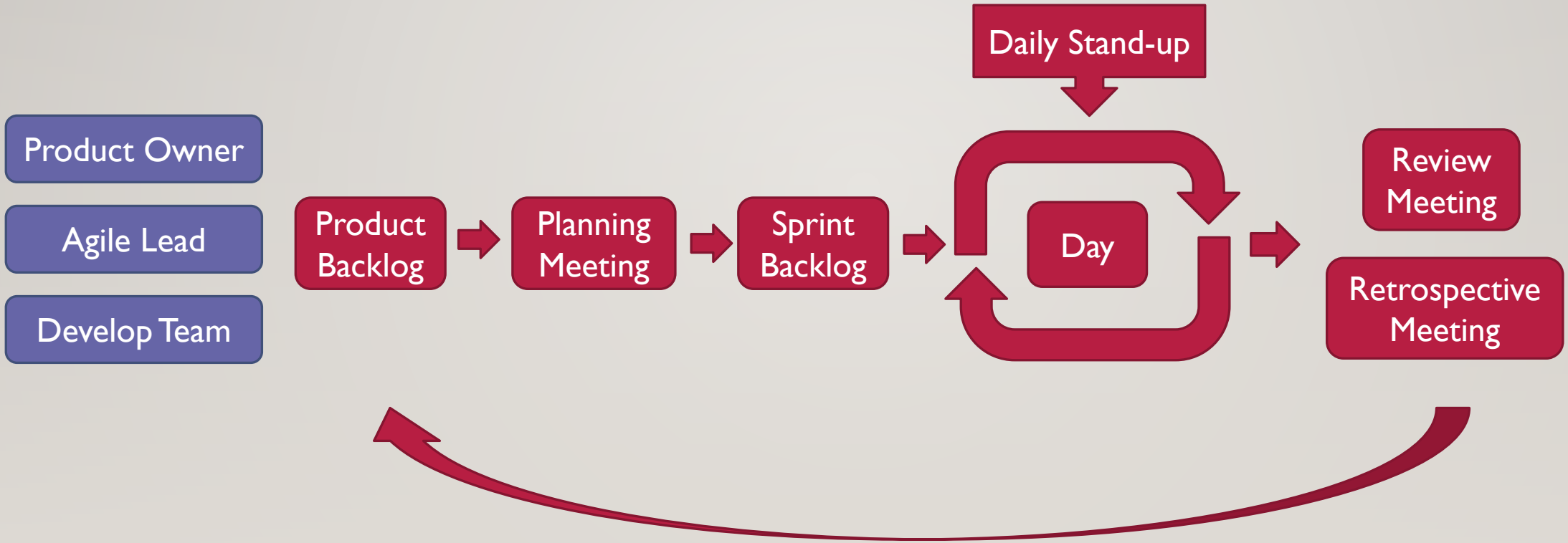
SCRUM WORKFLOW



FIVE EVENTS

- ***Sprint***: Sprint itself is an event, it composed of the following part
- ***Sprint Planning Meeting***: Planning for the upcoming Sprint. Including narrow down the task range and assign the tasks to team members.
- ***Daily Stand Up***: Daily report or your personal progress.
- ***Sprint Review Meeting***: Review the completeness of the tasks in this sprint.
- ***Sprint Retrospective Meeting***: Look back at the tasks, self evaluate and make improvement.

SCRUM WORKFLOW



BACKLOG MANAGEMENT

- To manage backlog means we need to store our application information in certain format: word doc, Google doc, pdf, etc.
- Software development tool – ***Jira***
- ***Ticket*** – Each task is called a ticket
  - Task type
  - Description
  - Comments

JIRA

- Example

DEVELOPMENT ENVIRONMENT

- Software development usually have many different environment:
  - ***Local***: application deployed to local machine, laptop/desktop, for local testing
  - ***Staging***: deployed to remote cloud service, testing in real environment; for internally use only.
  - ***Production***: live version, customers will use this endpoint. Everything should be tested.
  - ***Pre-Production***: also for internally use, but environment is pretty much similar to production.

DEVELOPMENT ENVIRONMENT

- Software development usually have many different environment:
  - ***Local***: localhost:8080
  - ***Staging***: www.testing.ebay.com
  - ***Production***: www.ebay.com
  - ***Pre-Production***: www.preprod.ebay.com

JENKINS

- Open source automation server
- Enable developers to reliably build, test and deploy software application
- Used by QE to do automation job
- Also used as part of CI/CD, generate code manifest

QUESTIONS