Homework

Функцията *sasho_formating(string, width)* приема непразен низ от символи string и естесвено число width, и го обработва по следния начин:

1. Премахва интервалите в началото и края на низа, ако има такива.

2. Редуцира всяка последователност от два или повече интервала между думите до един-единствен.

3. Заменя всяка малка буква с главна.

4. Центрира текста в поле с ширина width - допълва го поравно и от двете страни с интервали, така че дължината на низа да бъде равна на width. Това допълване се прави само ако дължината на текста след първите три стъпки е по-малка от параметъра width. Допълнително, ако

(width - дължината_на_низа_преди_центрирането)

е нечетно число, "нечетният" интервал трябва да се добави в края на низа.

Напишете колкото можете positive test cases(unit tests) в следния формат

sasho_formating("run SAshoo   run!!", 20) => " RUN SASHOO RUN!! "

3. Software Development Life-Cycle

How do people make software

Astea Solutions QA Team

Questions

- Why do we test?
- When we have a bug?
- What is test case?
- What is test suite?
- Test case types?

Overview

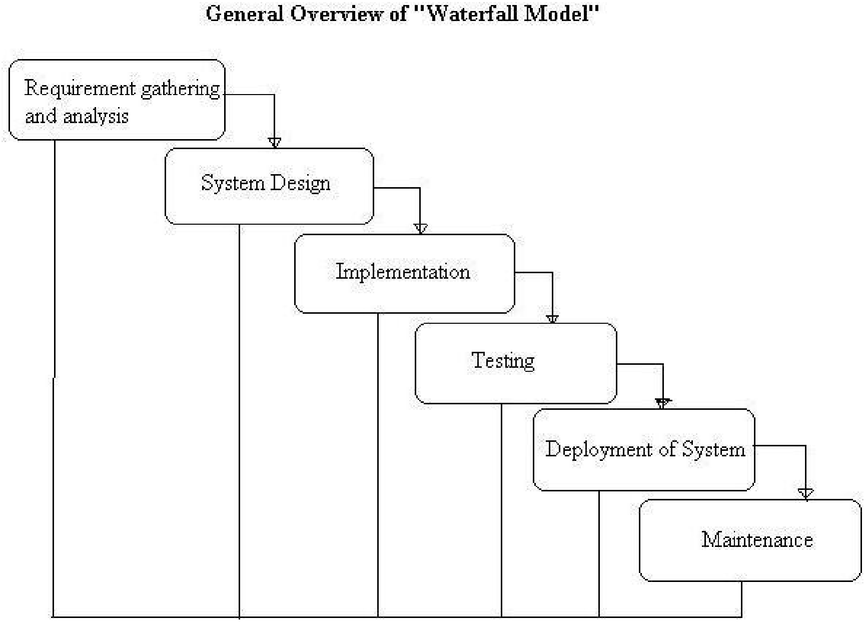
- Phases in software development
- Waterfall
- V-model
  - Component/Unit testing
  - Integration testing
  - System/End-to-end testing
  - Acceptance testing
- Agile methodologies
  - Scrum
  - Kanban

Phases in software development

- Requirement gathering and analysis
- Design
- Development
- Testing and bug fixing
- Release/Deployment
- Maintenance

Waterfall (1)

- Final product is developed in one linear iteration
- Each phase must be completed fully before the next phase can begin

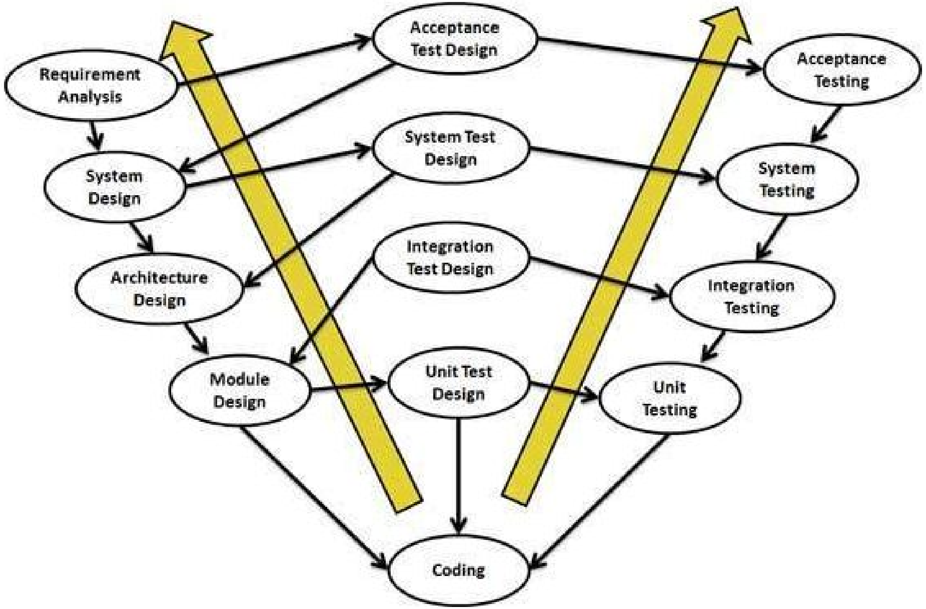

General Overview of "Waterfall Model"
Requirement gathering and analysis
System Design
Implementation
Testing
Deployment of System
Maintenance

Waterfall (2)

- Advantages
  - Simple and very well known
  - Easy to manage
  - Phases do not overlap
- Disadvantages
  - No working software is produced until late during the life cycle
  - Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
  - High amounts of risk and uncertainty - time and cost overrun
  - Not suitable for the projects where requirements are at a moderate to high risk of changing
- When to use
  - When the requirements are very well known, clear and fixed
  - Technology is understood

V-model (1)

- Main idea - development and testing tasks are corresponding activities of equal importance
- System is gradually being designed and finally programmed and tested
- Verification - Static method for verifying design and code
- Validation - Dynamic process for checking and testing the real product

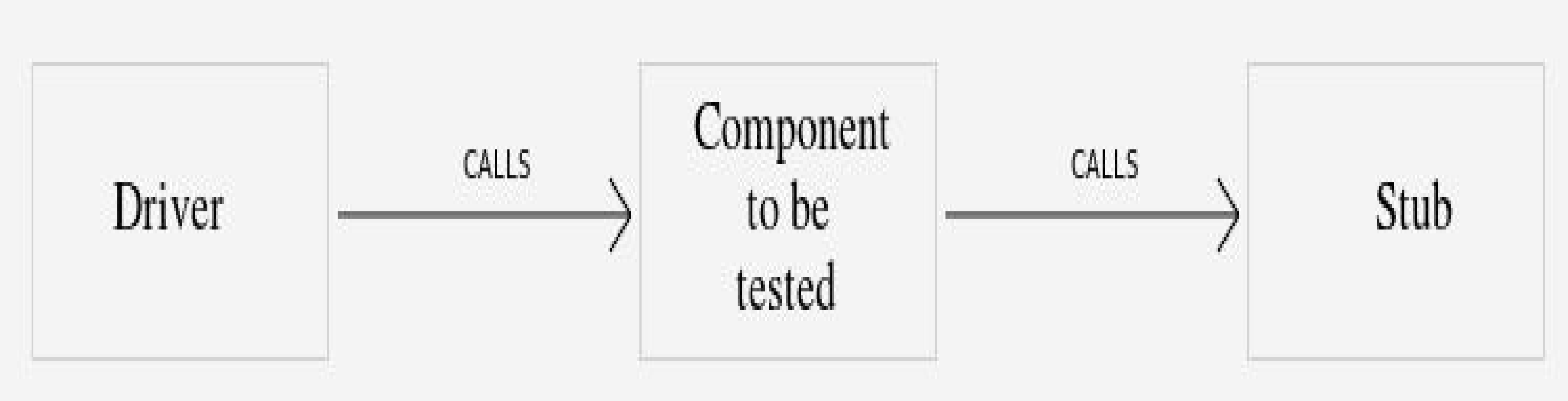


V-model (2)

- Component/Unit testing (1)
  - Typically occurs with access to the code being tested
  - In practice usually involves the programmer who wrote the code
  - Defects are typically fixed as soon as they found, without formally managing these defects
  - The disadvantages is that a programmer is testing his own program

V-model (3)

- Component/Unit testing (2)
    - Stubs - if the module/function A you are testing calls another module/function B, which is not ready, then use a simplified version of module/function B, called a stub
    - Drivers - allows you to call a module/function and display its return values



V-model (4)

- Integration testing
  - Tests the interactions between software components and is done after component testing
  - The greater the scope of integration, the more difficult it becomes to isolate defects to a specific component or system, which may lead to increased risk and additional time for troubleshooting

V-model (5)

- System/End-to-end testing
  - Checks if the integrated product meets the specified requirements
  - In the lower test levels, the testing was done against technical specifications. The system test, though, looks at the system from the perspective of the customer and the future user.
  - Many functions and system characteristics result from the interaction of all system components; consequently, they are visible only when the entire system is present and can be observed and tested only there
- System integration testing tests the interactions between different systems or between hardware and software and may be done after system testing

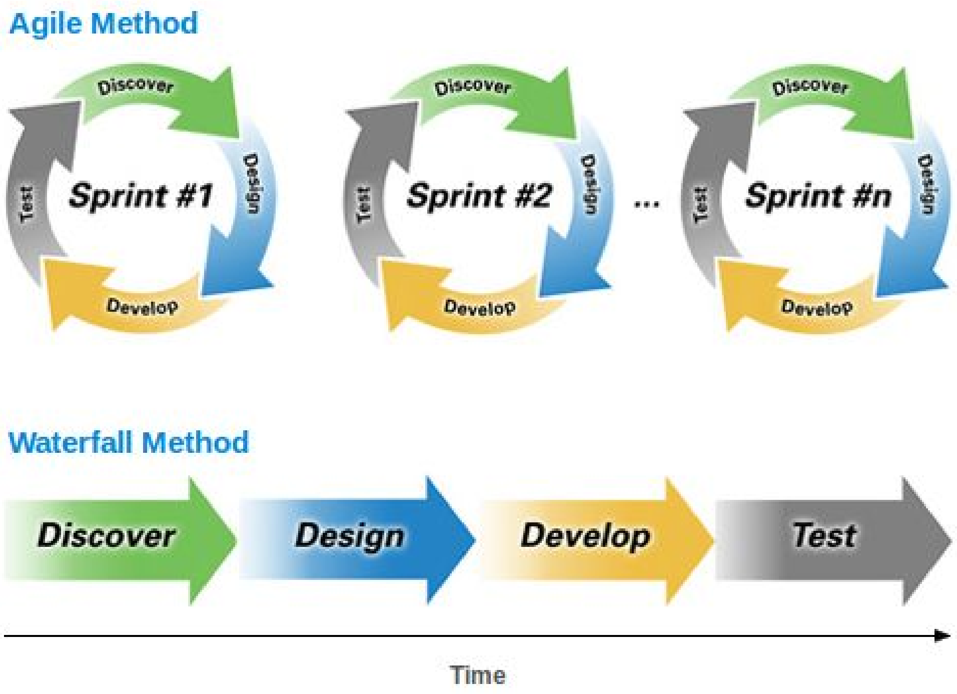
V-model (6)

- Acceptance testing
  - The goal is to establish confidence in the system
  - Assess the system’s readiness for deployment and use
  - Finding defects is not the main focus
  - How much acceptance testing is necessary depends on product risk
  - Customers, users, system administrators are involved
  - Acceptance tests may also be executed as a part of lower test levels or be distributed over several test levels:
    - Usability of a component can be acceptance tested during its component test.
    - Acceptance of new functionality can be checked on prototypes before system testing.

V-model (7)

- Advantages
  - In theory more reliable product than waterfall developed application
  - Simple and easy to use
  - Easy to manage
  - Phases do not overlap
- Disadvantages
  - No working software is produced until late during the life cycle
  - Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
  - High amounts of risk and uncertainty - time and cost overrun
  - Not suitable for the projects where requirements are at a moderate to high risk of changing
- When to use
  - When the requirements are very well known, clear and fixed
  - Technology is understood

Agile methodologies



Agile methodologies

- Advantages
  - Is a very realistic approach to software development
  - Suitable for fixed or changing requirements
  - Good model for environments that change steadily
  - Gives flexibility to developers
  - Customer satisfaction by rapid, continuous delivery of useful software
  - Continuous attention to technical excellence and good design.
- Disadvantages
  - The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
  - Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources
- When to use
  - Requirements are changing over software development life-cycle

Scrum (1)

- Roles
  - Product owner
  - Scrum master
  - Team
- Artefacts
  - Product backlog
  - Sprint backlog
  - Burndown charts
- Ceremonies
  - Sprint planning
  - Sprint review/demo
  - Sprint retrospective
  - Daily scrum meeting

Scrum (2)

- Sprint/Iteration - regular, repeatable work cycle. During each sprint, a team creates a potentially shippable product increment
- Product owner
  - Defines the features of the product
  - Decides on release date and content
  - Responsible for the profitability of the product (ROI)
  - Prioritizes features according to market value
  - Adjusts features and priority every iteration
  - Accepts or rejects work results

Scrum (3)

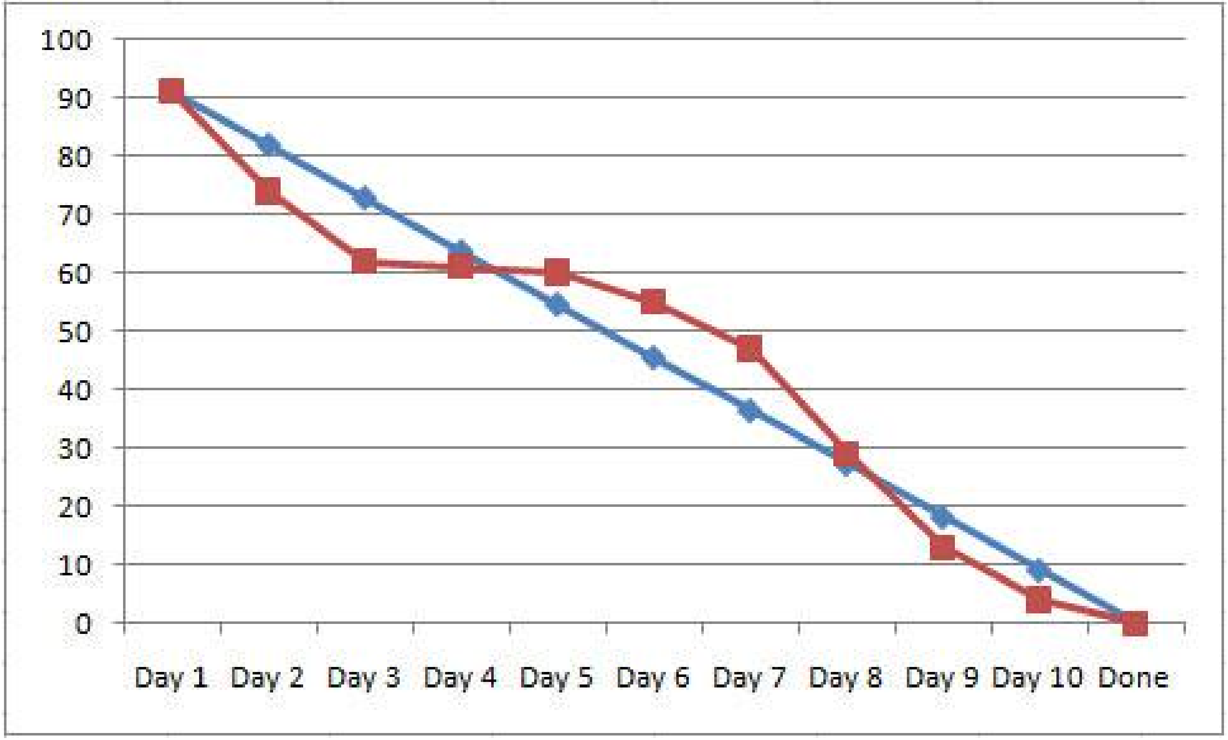
- Scrum master
  - Deeply understands the work being done by the team
  - Observing that the team obeys the rules and realizes the method of Scrum entirely
  - Does not interfere into the decisions of the team regarding specifically the development, but rather is there for the team as an advisor
  - Gives only impulses and advises to the team to lead the correct way, to use the right method or to choose the right technology
- Team
  - Cross-functional - software engineers, architects, programmers, analysts, QA experts, testers, UI designers, etc.
  - Decides self dependent, which requirements or User Stories it can accomplish in one sprint

Scrum (4)

- Product backlog
  - The list of functionality, technology and issues
  - Managed and prioritised by Product owner
  - Emerging, prioritised, estimated
  - One list for multiple teams on product
- Sprint backlog
  - List of tasks the team needs to address during the sprint
  - Tasks are estimated by the team
  - Team members sign up for tasks, they are not assigned
  - Estimated work remaining is updated daily
  - Only team can change it

Scrum (5)

- Burndown charts



Scrum (5)

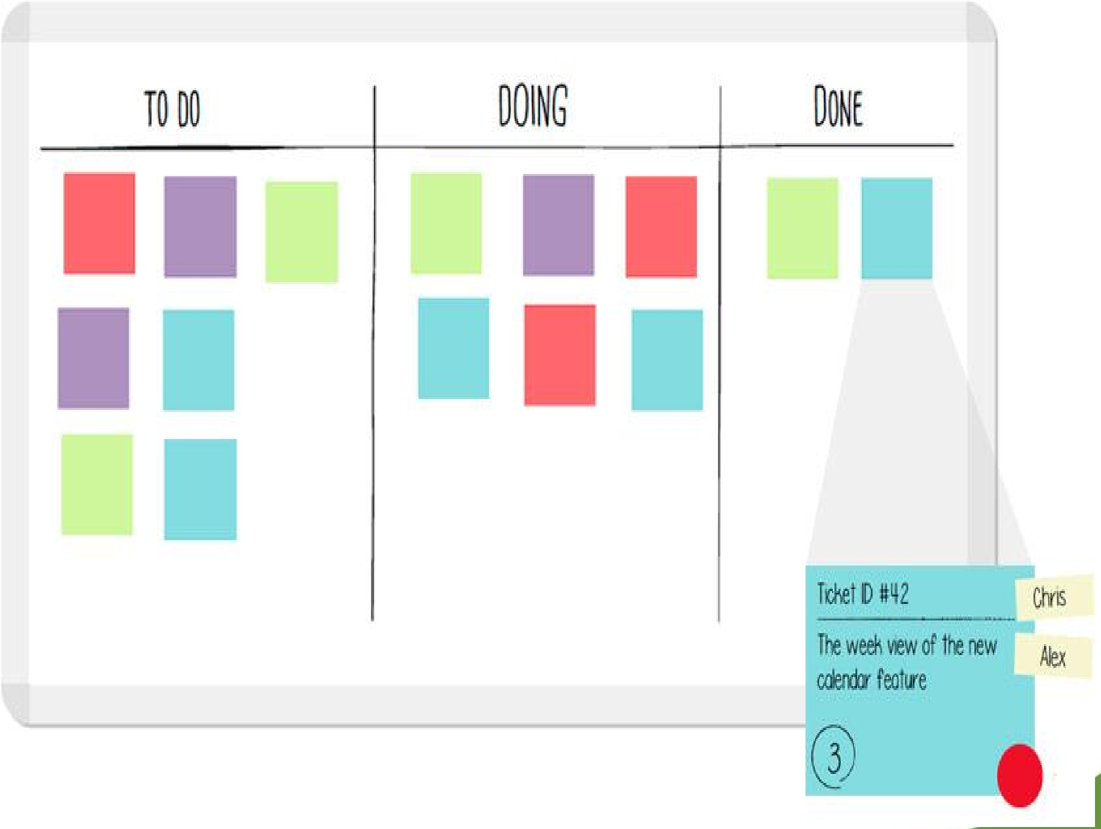
- Sprint planning
  - Team picks items from product backlog that they can commit to complete
  - Sprint backlog is created
    - List of tasks necessary to achieve the work
    - Task are identified and each item is estimated
    - Scrum master does NOT decide for the team
  - Team self-organises to meet the goal - tasks are NOT assigned by manager
- Daily standup
  - Happens everyday at a fixed time
  - 15 minutes long stand up meeting
  - 3 questions are answered by every team member
    - What did I do yesterday?
    - What do I plan to do today?
    - Do I have some blocker?
  - Only one team member can speak at a time
  - Specific issues are resolved offline

Scrum (6)

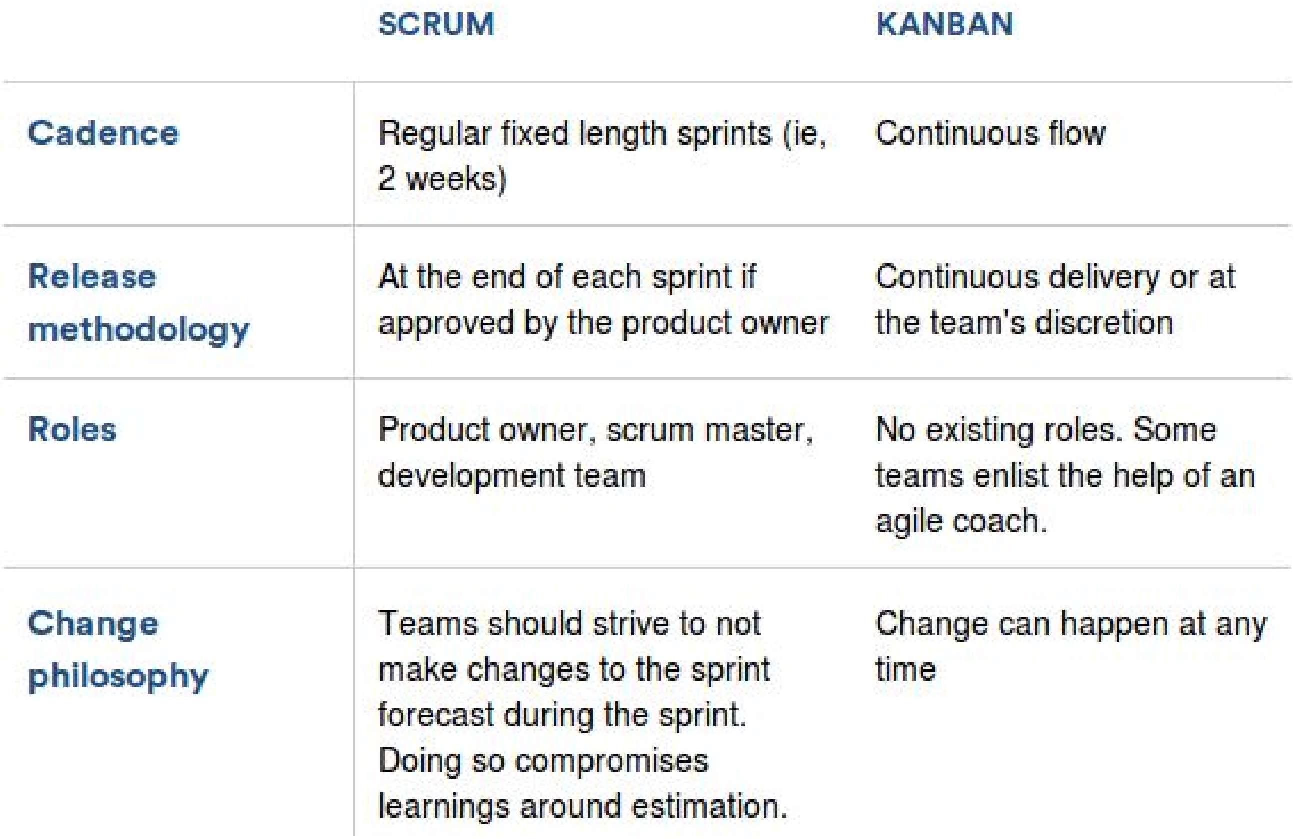
- Sprint review
  - Informal
  - Team presents what is done during the sprint
  - Typically takes the form of a demo of new features or underlying architecture
  - Whole team participates
- Sprint retrospective
  - Review at what is working and is not for the team
  - Issues must be acted upon
  - Typically an hour or so
  - Done after every sprint
  - Whole team participates

Kanban (1)

- A kanban team is only focused on the work that's actively in progress
- The product owner is free to re-prioritize work in the backlog
- Flexibility in planning
- Minimizing cycle time
- Moving toward continuous delivery - the practice of building and validating code incrementally throughout the day–is essential for maintaining quality



Kanban (2)



	SCRUM	KANBAN
Cadence	Regular fixed length sprints (ie, 2 weeks)	Continuous flow
Release methodology	At the end of each sprint if approved by the product owner	Continuous delivery or at the team's discretion
Roles	Product owner, scrum master, development team	No existing roles. Some teams enlist the help of an agile coach.
Change philosophy	Teams should strive to not make changes to the sprint forecast during the sprint. Doing so compromises learnings around estimation.	Change can happen at any time

Summary

- What are the phases in software development life-cycle?
- What is end-to-end testing?
- What are the scrum roles?
- What are the scrum artefacts?
- What is sprint in scrum?
- What are the questions in daily standup that every team member should answer?

QUESTIONS