**Notes for MCA-I (Semester- I)**

**Subject :- Object Oriented Software Engineering**

**(Subject Code:- IT-13)**

**Chapter: 5**] **Current Trends in Software Engineering**

* **5.1 Introduction to Web Engineering :-**

**Web engineering** focuses on the methodologies, techniques, and tools that are the foundation of [Web application development](https://en.wikipedia.org/wiki/Web_application_development) and which support their design, development, evolution, and evaluation. Web application development has certain characteristics that make it different from traditional software, information system, or computer application development.

Web engineering is multidisciplinary and encompasses contributions from diverse areas: [systems analysis](https://en.wikipedia.org/wiki/Systems_analysis) and [design](https://en.wikipedia.org/wiki/Systems_design), software engineering, hypermedia/hypertext engineering, [requirements engineering](https://en.wikipedia.org/wiki/Requirements_engineering), [human-computer interaction](https://en.wikipedia.org/wiki/Human-computer_interaction), user interface, [information engineering](https://en.wikipedia.org/wiki/Information_engineering), information indexing and [retrieval](https://en.wikipedia.org/wiki/Information_retrieval), testing, modelling and simulation, project management, and graphic design and presentation. Web engineering is neither a clone nor a subset of software engineering, although both involve programming and software development. While Web Engineering uses software engineering principles, it encompasses new approaches, methodologies, tools, techniques, and guidelines to meet the unique requirements of [Web-based applications](https://en.wikipedia.org/wiki/Web_application).

**Software Engineering** is defined as the application of science and mathematics by which the capabilities of computer equipment are made useful to man via computer programs, procedures, and associated documentation.

**Web Engineering** is the application of systematic and quantifiable approaches (concepts methods, techniques tools) to cost ‐ effective requirements analysis, design, implementation, testing, operation, and maintenance of high ‐quality Web applications.

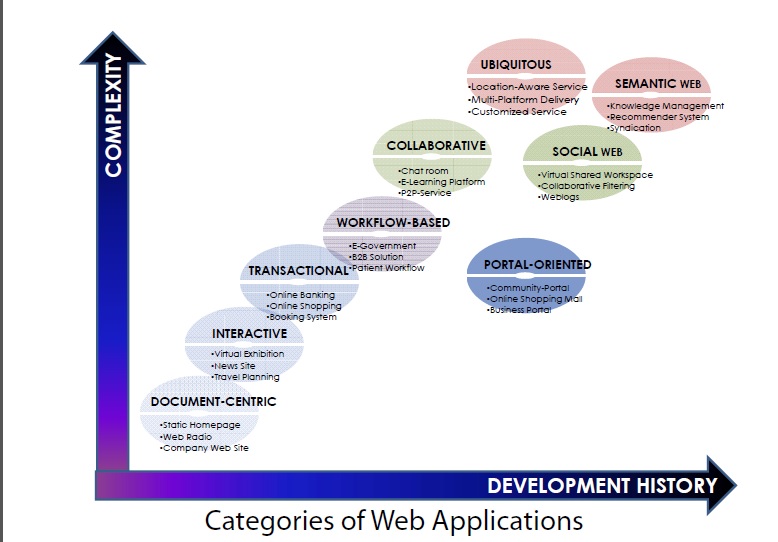
**WebApps**

• The term *Web application* (WebApp) encompasses:

– everything from a simple Web page that might help a consumer compute an automobile lease payment to a comprehensive website that provides complete travel services for business people and vacationers.

– Included within this category are complete websites, specialized functionality within websites, and information processing applications that reside on the Internet or on an Intranet or Extranet.

|  |  |
| --- | --- |
| **WebApp Attributes :-** | **WebApp Types :-** |
| • Network intensiveness | • Informational |
| • Concurrency | • Transaction‐oriented |
| • Content sensitive | • Download |
| • Continuous evolution | Customizable |
| • Unpredictable load | • Service‐oriented |
| • Immediacy | • Interaction |
| • Performance | • Portals |
| • Security | • Database Access |
| • Availability | • User input |
| • Data driven | • Data warehousing |
| • Aesthetics |  |

****

**Why Web Engineering?**

**• As WebApps become larger and more complex,**

**–** Informality remains, but some degree of requirements gathering and planning are necessary

– Urgency remains, but it must be tempered by a recognition that decisions may have broad consequences

– Intuition remains, but it must be augmented by proven management and technical patterns

– Art remains, but it must be complemented with solid design

Proponents of Web engineering supported the establishment of Web engineering as a discipline at an early stage of Web. Major arguments for Web engineering as a new discipline are:

* Web-based Information Systems (WIS) development process is different and unique.
* Web engineering is multi-disciplinary; no single discipline (such as software engineering) can provide complete theory basis, body of knowledge and practices to guide WIS development.
* Issues of evolution and lifecycle management when compared to more 'traditional' applications.
* Web-based information systems and applications are pervasive (All encompassing) and non-trivial. The prospect of Web as a platform will continue to grow and it is worth being treated specifically.

However, it has been controversial, especially for people in other traditional disciplines such as software engineering, to recognize Web engineering as a new field. The issue is how different and independent Web engineering is, compared with other disciplines.

Main topics of Web engineering include, but are not limited to, the following areas:

**Modeling disciplines**

* Business Processes for Applications on the Web
* Process Modelling of Web applications
* Requirements Engineering for Web applications
* B2B applications

**Design disciplines, tools, and methods**

* [UML](https://en.wikipedia.org/wiki/Unified_Modeling_Language) and the Web
* Conceptual Modeling of Web Applications (aka. [Web modeling](https://en.wikipedia.org/wiki/Web_modeling))
* Prototyping Methods and Tools
* [Web design](https://en.wikipedia.org/wiki/Web_design) methods
* CASE Tools for Web Applications
* Web Interface Design
* Data Models for Web Information Systems

**Implementation disciplines**

* Integrated Web Application Development Environments
* Code Generation for Web Applications
* Software Factories for/on the Web
* Web 2.0, AJAX, E4X, ASP.NET, PHP and Other New Developments
* [Web Services](https://en.wikipedia.org/wiki/Web_Service) Development and Deployment

**Testing disciplines**

* Testing and Evaluation of Web systems and Applications.
* Testing Automation, Methods, and Tools.

**Applications categories disciplines**

* [Semantic (logic based) Web](https://en.wikipedia.org/wiki/Semantic_Web) applications
* Document centric Web sites
* Transactional Web applications
* Interactive Web applications
* Workflow-based Web applications
* Collaborative Web applications
* Portal-oriented Web applications
* Ubiquitous (ever present or everywhere )and Mobile Web Applications
* Device Independent Web Delivery
* [Localization and Internationalization](https://en.wikipedia.org/wiki/Internationalization_and_localization) of Web Applications
* [Personalization](https://en.wikipedia.org/wiki/Personalization) of Web Applications

**Web quality**

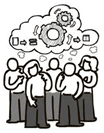
* Web Metrics, Cost Estimation, and Measurement
* Personalization and Adaptation of Web applications
* Web Quality
* [Usability of Web Applications](https://en.wikipedia.org/wiki/Web_usability)
* [Web accessibility](https://en.wikipedia.org/wiki/Web_accessibility)
* Performance of Web-based applications

**Content-related**

* Web Content Management
* Content Management System (CMS)
* [Multimedia](https://en.wikipedia.org/wiki/Multimedia) Authoring Tools and Software
* [Authoring of adaptive hypermedia](https://en.wikipedia.org/wiki/Authoring_of_adaptive_hypermedia)
* **5.2 Agile Process :-**

Agile is a software development methodology to build a software incrementally using short iterations of 1 to 4 weeks so that the development is aligned with the changing business needs.

AGILE methodology is a practice that promotes **continuous iteration** of development and testing throughout the software development lifecycle of the project. In the Agile model, both development and testing activities are concurrent, unlike the Waterfall model.

[](https://www.guru99.com/images/11-2014/agile_Processesv1_1.png)

Agile Methodology

An agile methodology is an iterative approach to software development. Each iteration of agile methodology takes a short time interval of 1 to 4 weeks. The agile development process is aligned to deliver the changing business requirement. It distributes the software with faster and fewer changes.

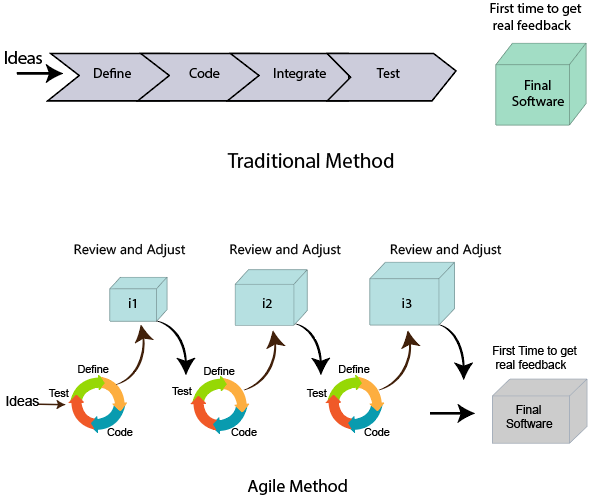
The single-phase software development takes 6 to 18 months. In single-phase development, all the requirement gathering and risks management factors are predicted initially.

The agile software development process frequently takes the feedback of workable product. The workable product is delivered within 1 to 4 weeks of iteration.

**Agile Model Vs Waterfall Model**

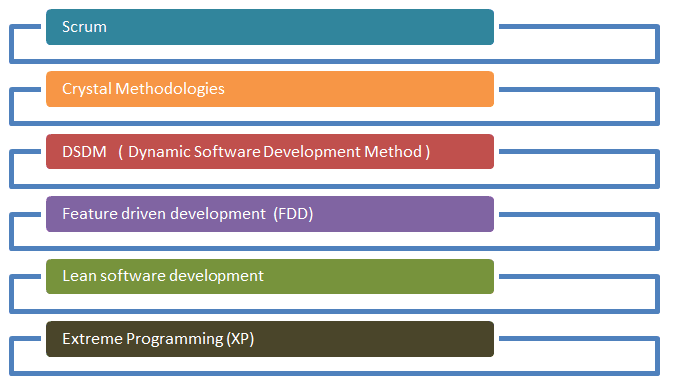
Agile and Waterfall model are two different methods for software development process. Though they are different in their approach, both methods are useful at times, depending on the requirement and the type of the project.

|  |  |
| --- | --- |
| **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 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. |
| * 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 |

****

**Agile Process**

Check below Agile process model to deliver successful systems quickly.

[](https://www.guru99.com/images/11-2014/agile_Processesv1_2.png)

**Roles in Agile**

There are two different roles in a Agile methodology. These are the Scrum Master and Product Owner.

**1. Scrum Master**

The Scrum Master is a team leader and facility provider who helps the team member to follow agile practices, so that the team member meets their commitments and customers requirements. The scrum master plays the following responsibilities:

* They enable the close co-operation between all the roles and functions.
* They remove all the blocks which occur.
* They safeguard the team from any disturbances.
* They work with the organization to track the progress and processes of the company.
* They ensure that Agile Inspect & Adapt processes are leveraged correctly which includes
  + Planned meetings
  + Daily stand-ups
  + Demo
  + Review
  + Retrospective meetings, and
  + Facilitate team meetings and decision-making process.

**2. Product Owner**

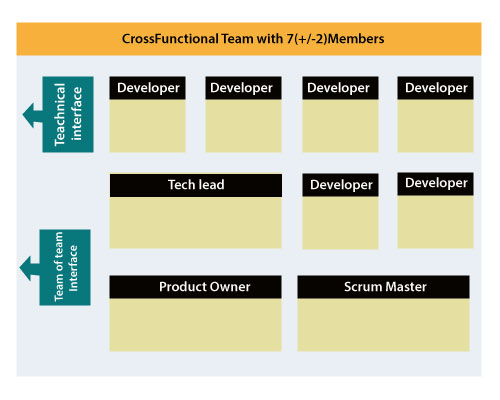
The Product Owner is one who runs the product from a business perspective. The Product Owner plays the following responsibilities:

* He defines the requirements and prioritizes their values.
* He sets the release date and contents.
* He takes an active role in iteration and releasing planning meetings.
* He ensures that the team is working on the most valued requirement.
* He represents the voice of the customer.
* He accepts the user stories that meet the definition of done and defined acceptance criteria.

**Cross-functional team:-**

Every agile team contains self-sufficient team with 5 to 9 team members. The average experience of each member ranges from 6 to 10 years. The agile team contains 3 to 4 developers, 1 tester, 1 technical lead, 1 scrum master and 1 product owner.

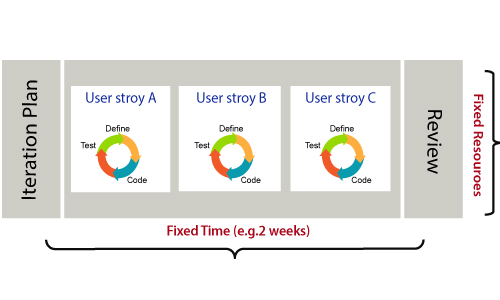
The Scrum master and Product owner are considered as a part of Team Interface, on the other hand remaining members are the part of Technical Interface.

****

**How an Agile Team plan their work?**

An Agile methodology is not a specific set of ceremonies or specific development techniques. Rather, it is a group of methodologies that demonstrate a commitment to tight feedback cycles and continuous improvement. An Agile team works in iterations to deliver the customer requirement, and each iteration takes 10 to 15 days. However, the original Agile Manifesto didn't set the time period of two-week iterations or an ideal team size.

Each user requirement is a planned based and their backlog prioritization and size. The team decides, how much scope they have and how many hours available with each team to perform their planed task.

****

**What is a user requirement?**

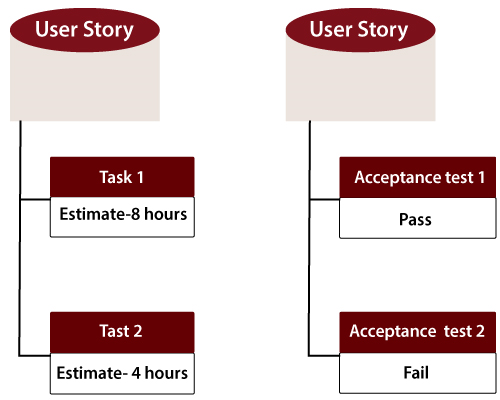
The user requirement defines the requirements of the user in terms of functionalities. There may be of two type of functionality.

* As a <User Role> I want <Functionality> so that <Business Value>
* In order to <Business value> as a <User Role> I want <Functionality>.

During software release planning, a rough estimate is given to a user requirement using relative scale points. During iteration planning, the requirement is broken down into tasks.

**Relation between User requirement and Task**

* User requirement talks about what is to be done. It defines the needs of users.
* Task talks about how it is to be done. It defines how functionality is implemented.
* User requirements are implemented by tasks. Every requirement is gathering as the task.
* User requirement is divided into different tasks when it is planned in current iteration.
* User tasks are estimated in hours based, generally it is between 2 to 12 hours.
* Requirements are validated using acceptance test.

****

**When the requirement is completed**

The Agile team decides the meaning of task done. There may be different criteria for it:

* When the entire task (development, testing) are completed.
* When all the acceptance tests are running and are passed.
* When no defects found.
* Product owner has accepted the requirement.
* When the software product is delivered to the end user.

**What is Software Acceptance Criteria?**

Acceptance Criteria is defined as the functionality, behavior, and performance required by a product owner. It defines what is to be done so that the developer knows when a user requirement is complete.

**How the Requirements are Defined?**

Requirements are defined as

* A User Story,
* With Acceptance Criteria, and
* Tasks to implement the story.

**Twelve Principles of Agile Manifesto :-**

* **Customer Satisfaction** − Highest priority is given to satisfy the requirements of customers through early and continuous delivery of valuable software.
* **Welcome Change** − Changes are inevitable during software development. Ever-changing requirements should be welcome, even late in the development phase. Agile processes should work to increase customers' competitive advantage.
* **Deliver a Working Software** − Deliver a working software frequently, ranging from a few weeks to a few months, considering shorter time-scale.
* **Collaboration** − Business people and developers must work together during the entire life of a project.
* **Motivation** − Projects should be built around motivated individuals. Provide an environment to support individual team members and trust them so as to make them feel responsible to get the job done.
* **Face-to-face Conversation** − Face-to-face conversation is the most efficient and effective method of conveying information to and within a development team.
* **Measure the Progress as per the Working Software** − Working software is the key and it should be the primary measure of progress.
* **Maintain Constant Pace** − Agile processes aim towards sustainable development. The business, the developers, and the users should be able to maintain a constant pace with the project.
* **Monitoring** − Pay regular attention to technical excellence and good design to enhance agility.
* **Simplicity** − Keep things simple and use simple terms to measure the work that is not completed.
* **Self-organized Teams** − An agile team should be self-organized and should not depend heavily on other teams because the best architectures, requirements, and designs emerge from self-organized teams.
* **Review the Work Regularly** − Review the work done at regular intervals so that the team can reflect on how to become more effective and adjust its behavior accordingly.

**Agile Characteristics :-**

**Iterative/incremental and Ready to Evolve**

Most of the agile development methods break a problem into smaller tasks. There is no direct long-term planning for any requirement. Normally, iterations are planned which are of vary short period of time, for example, 1 to 4 weeks. A cross-functional team is created for each iteration that works in all functions of software development like planning, requirements analysis, design, coding, unit testing, and acceptance testing. The result at the end of the iteration is a working product and it is demonstrated to the stakeholders at the end of an iteration.

After demo, review comments are taken and are planned to be incorporated in the working software as required.

**Face-to-face Communication**

Each agile team should have a customer representative such as a product owner in scrum methodology. This representative is authorized to act on behalf of the stakeholders and he can answer the queries of the developers in between iterations.

An information radiator (physical display) is normally located prominently in an office, where passers-by can see the progress of the agile team. This information radiator shows an up-to-date summary of the status of a project.

**Feedback Loop**

Daily stand-up is a common culture of any agile development; it is also known as **daily scrum**. It is a kind of a brief session where each team member reports to each other regarding the status of what they have done, what to do next, and any issues they are facing.

* **Agile Daily Stand Up**

Daily stand-up, as the name suggests, is a daily status meeting among all the members of an agile team. It not only provides a forum for regular updates but also brings the problems of team members into focus so that it can be quickly addressed. Daily stand-up is a must-do practice, no matter how an agile team is established regardless of its office location.

**What is Daily Stand-up?**

* A daily stand-up is a daily status meeting among all team members and it is held roughly for 15 minutes.
* Every member has to answer three important questions −
  + What I did yesterday?
  + What I'll do today?
  + Any impediment I am facing.../ I am blocked due to...
* Daily stand-up is for status update, not for any discussion. For discussion, team members should schedule another meeting at a different time.
* Participants usually stand instead of sitting so that the meeting gets over quickly.

**Why Stand-up is Important?**

The benefits of having a daily stand-up in agile are as follows −

* The team can evaluate the progress on a daily basis and see if they can deliver as per the iteration plan.
* Each team member informs all about his/ her commitments for the day.
* It provides visibility to the team on any delay or obstacles.

**Who Attends a Stand-up?**

* The scrum master, the product owner, and the delivery team should attend the stand-up on a daily basis.
* Stakeholders and Customers are encouraged to attend the meeting and they can act as an observer, but they are not supposed to participate in stand-ups.
* It is the scrum master's responsibility to take note of each team member's queries and the problems they are facing.

**Geographically Dispersed Teams**

Stand-ups can be done in multiple ways, in case the agile team members are operating from different time zones −

* Select a member on a rotational basis, who can attend the stand-up meeting of teams located in different time zones.
* Have a separate stand-up per team, update the status of the stand-up in a tool such as Rally, SharePoint, Wikis, etc.
* Have a wide variety of communication tools ready like conference call, video conferencing, instant messengers, or any other third-party knowledge sharing tools.

The definition of **done** for User Story, Iteration, and Release is given below.

**User Story**

A user story is a requirement which is formulated in a few sentences in everyday language of an user and it should be completed within an iteration. A user story is done when

* All the related code have been checked-in.
* All the unit test cases have been passed.
* All the acceptance test cases have been passed.
* Help text is written.
* Product Owner has accepted the story.

**Iteration**

An iteration is a time boxed collection of user stories / defects to be worked upon and accepted within the release of a product. Iterations are defined during iteration planning meeting and completed with an iteration demo and review meeting. An iteration is also termed as a **sprint**. An iteration is done when

* Product backup is complete.
* Performance has been tested.
* User stories have been accepted or moved to the next iteration.
* Defects have been fixed or postponed to the next iteration.

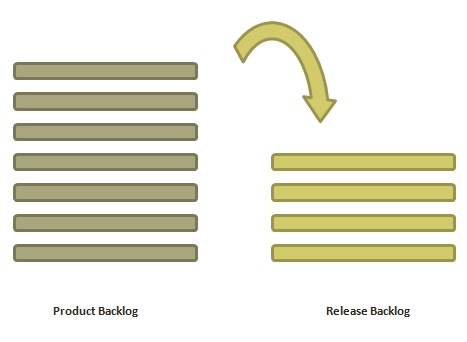
**Release**

A release is a major milestone that represents an internal or external delivery of working, tested version of the product/system. A release is done when

* System is stress tested.
* Performance is tuned.
* Security validations are carried out.
* Disaster recovery plan is tested.

# Agile - Release Planning

The purpose of release planning is to create a plan to deliver an increment to the product. It is done after every 2 to 3 months.



**Who is Involved?**

* **Scrum Master** − The scrum master acts as a facilitator for the agile delivery team.
* **Product Owner** − The product owner represents the general view of the product backlog.
* **Agile Team** − Agile delivery team provides insights on the technical feasibilities or any dependencies.
* **Stakeholders** − Stakeholders like customers, program managers, subject matter experts act as advisers as decisions are made around the release planning.

**Prerequisites of Release Planning**

The prerequisites of release planning are as follows −

* A ranked product backlog, managed by the Product Owner. Generally five to ten features are taken which the product owner feels that can be included in a release
* Team's input about capabilities, known velocity or about any technical challenge
* High-level vision
* Market and Business objective
* Acknowledgement whether new product backlog items are needed

**Materials Required**

The list of materials required for release planning is as follows −

* Posted agenda, purpose
* Flip charts, whiteboards, markers
* Projector, way to share computers having data/tools required during planning meeting
* Planning data

**Planning Data**

The list of data required to do release planning is as follows −

* Previous iterations or release planning results
* Feedback from various stakeholders on product, market conditions, and deadlines
* Action plans of previous releases / iterations
* Features or defects to be considered
* Velocity from previous releases/ estimates.
* Organizational and personal calendars
* Inputs from other teams and subject matter experts to manage any dependencies

**Output**

The output of a release planning can be the following −

* Release plan
* Commitment
* Issues, concerns, dependencies, and assumptions which are to be monitored
* Suggestions to improve future release planning

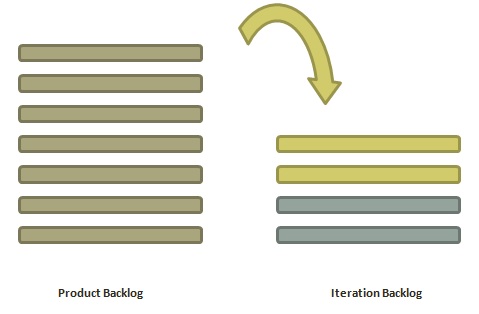
**Agenda**

The agenda of a release planning can be −

* **Opening ceremony** − Welcome message, review purpose and agenda, organizing tools and introduction to business sponsors.
* **Product Vision, Roadmap** − Show the large picture of the product.
* **Review previous releases** − Discussion on any item which can impact the plan.
* **Release name / theme** − Inspect the current status of roadmap themes and do the required adjustments, if any.
* **Velocity** − Present the velocity for the current release and of previous releases.
* **Release schedule** − Review key milestones and decision on time boxes for release and iterations within release.
* **Issues and concerns** − Check any concerns or issue and record them.
* **Review and Update the Definition of Done** − Review the definition of **done** and make appropriate changes based on technology, skill, or changes in team members since the last iteration / release.
* **Stories and items to be considered** − Present the user stories and features from the product backlog to be considered for scheduling in the current release.
* **Determine sizing values** − If the velocity is unknown, then plan the sizing values to be used in the release planning.
* **Coarse the size of stories** − The delivery team determines the appropriate size of the stories under consideration and splits the stories into multiple iterations if a story is too large. The product owner and the subject matter experts clarify the doubts, elaborate the acceptance criteria, and make proper story splits. The scrum master facilitates the collaboration.
* **Map stories to iterations** − The delivery team and the product owner move the stories/defects in the iterations based on the size and velocity. The scrum master facilitates the collaboration.
* **New concerns or issues** − Check any new issues based on previous experience and record the same.
* **Dependencies and assumptions** − Check any dependencies/assumptions planned during the release planning.
* **Commit** − The scrum master calls for the planning. Delivery team and Product owner signal it as the best plan and then commit to move to the next level of planning, that is, iteration planning.
* **Communication and logistics planning** − Review/Update the communication and logistics planning for the release.
* **Parking lot** − Process parking lot means all items should be either resolved or set as action items.
* **Distribute Action items and action plans** − Distribute the action items among their owners, process the action plan.
* **Retrospect** − Solicit feedback from participants to make the meeting successful.
* **Close** − Celebrate the success.

# Agile - Iteration Planning

The purpose of iteration planning is for the team to complete the set of top-ranked product backlog items. This commitment is time boxed based on the length of iteration and team velocity.



**Who is Involved?**

* **Scrum Master** − The scrum master acts as a facilitator for the agile delivery team.
* **Product Owner** − The product owner deals with the detailed view of the product backlog and their acceptance criteria.
* **Agile Team** − Agile delivery defines their tasks and sets the effort estimates required to fulfil the commitment.

**Prerequisites of Planning**

* Items in product backlog are sized and have a relative story point assigned.
* Ranking has been given to portfolio items by the product owner.
* Acceptance criteria has been clearly stated for each portfolio item.

**Planning Process**

Following are the steps involved in iteration planning −

* Determine how many stories can fit in an iteration.
* Break these stories into tasks and assign each task to their owners.
* Each task is given estimates in hours.
* These estimates help team members to check how many task hours each member have for the iteration.
* Team members are assigned tasks considering their velocity or capacity so that they are not overburdened.

**Velocity Calculation**

An agile team calculates velocity based on past iterations. Velocity is an average number of units required to finish user stories in an iteration. For example, if a team took 12, 14, 10 story points in each iteration for the last three iterations, the team can take 12 as velocity for the next iteration.

Planned velocity tells the team how many user stories can be completed in the current iteration. If the team quickly finishes the tasks assigned, then more user stories can be pulled in. Otherwise, stories can be moved out too to the next iteration.

**Task Capacity**

The capacity of a team is derived from the following three facts −

* Number of ideal working hours in a day
* Available days of person in the iteration
* Percentage of time a member is exclusively available for the team.

Suppose a team has 5 members, committed to work full time (8 hours a day) on a project and no one is on leave during an iteration, then the task capacity for a two-week iteration will be −

5 × 8 × 10 = 400 hours

**Planning Steps**

* Product Owner describes the highest ranked item of product backlog.
* Team describes the tasks required to complete the item.
* Team members own the tasks.
* Team members estimate the time to finish each task.
* These steps are repeated for all the items in the iteration.
* If any individual is overloaded with tasks, then his/her task is distributed among other team members.

**Agile Product Backlog**

The agile product backlog in [**Scrum**](https://www.javatpoint.com/agile-scrum)is a list of prioritized features. It contains a short description of all the functionalities desired in the product. In usual scenario, items should be broken down into user stories. Commonly, a Scrum team and its product owner write everything that they can think for agile backlog prioritization.

**Why Product Backlog is Important?**

* The backlog is prepared to provide an estimate of each feature.
* It helps in the planning of the product's roadmap.
* It helps in the re-ranking the features of the product by adding more value to it.
* It assists in determining the priority of the product first. The team member works first on the higher prioritize product.

**Characteristics of Product Backlog**

* Each product should have one product backlog which can have a set of large to very large features.
* Multiple teams can work on a single product backlog.
* Ranking of features is done based on business value, technical value, risk management or strategic fitness.
* Highest ranking items are decomposed into smaller stories during release planning so that they can be completed in future iterations

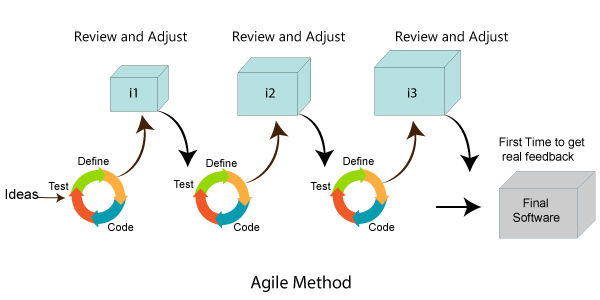
**The Product Backlog comprises the following different types of items:**

* Features
* Bugs
* Technical work
* Knowledge acquisition
* **Advantage of Agile Methodology**

There are various advantages of using agile methodology over traditional waterfall model or others. Agile development methodology and testing practices have worked wonders for numerous organizations with positive aspects. Its positive aspects are not hidden, it is very much visible in the organization.

**Advantages of Agile Methodology**

1. Customer satisfaction is rapid, continuous development and delivery of useful software.
2. Customer, Developer, and Product Owner interact regularly to emphasize rather than processes and tools.
3. Product is developed fast and frequently delivered (weeks rather than months.)
4. A face-to-face conversation is the best form of communication.
5. It continuously gave attention to technical excellence and good design.
6. Daily and close cooperation between business people and developers.
7. Regular adaptation to changing circumstances.
8. Even late changes in requirements are welcomed.

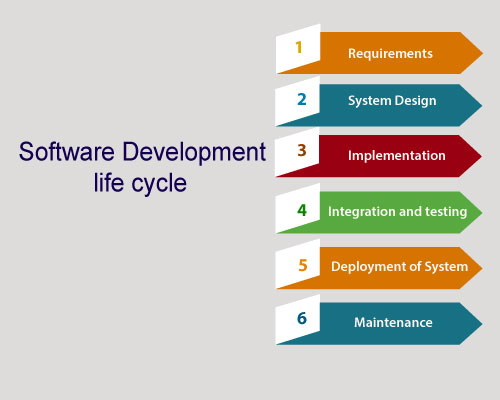


**Disadvantages of Agile methodology:**

1. It is not useful for small development projects.
2. There is a lack of intensity on necessary designing and documentation.
3. It requires an expert project member to take crucial decisions in the meeting.
4. Cost of Agile development methodology is slightly more as compared to other development methodology.
5. The project can quickly go out off track if the project manager is not clear about requirements and what outcome he/she wants.

**Advantages of the Waterfall Model:**

1. It is one of the easiest and traditional model to manage. Because of its traditional development nature, each phase has specific deliverables and a review process.
2. It works well in smaller size projects where requirements are easily understandable.
3. It has a faster product delivery model.
4. There are well-documented process and results.
5. Easily adaptable method for shifting teams
6. This project management methodology is beneficial to manage dependencies.



**Disadvantages of Waterfall Model:**

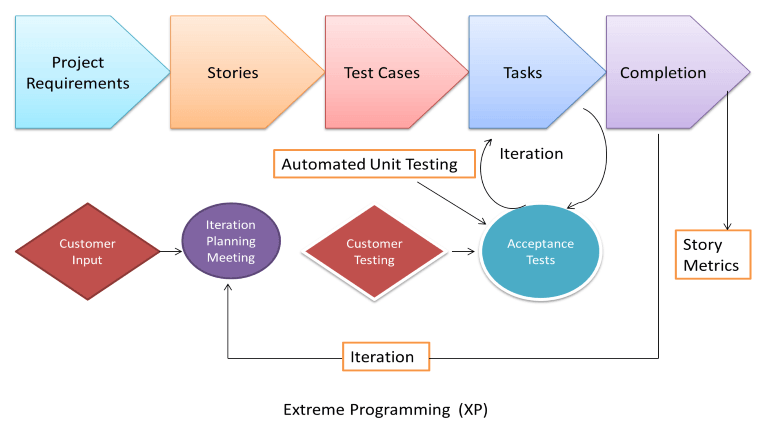
1. It is not an ideal model to develop a large scale project size.
2. It requires a clear-cut requirement at the beginning time; otherwise, it may lead to a less effective method.
3. It is difficult to move back to make changes in the previous phase.
4. The testing process starts once development is completed. Hence, it has high chances of bugs to be found later in project development. Due to this, it is costly to fix.

Comparison between the Agile methodology and Waterfall model:

|  |  |
| --- | --- |
| **Agile methodology** | **Waterfall model** |
| It follows the incremental approach. | It is a sequential design process. |
| It divides the project development lifecycle into a sprint. | The software development process is divided into distinct phases. |
| Agile methodology is a flexible methodology. | The Waterfall is a structured software development methodology. |
| Agile is the collection of many different projects. | It is completed as one single project. |
| The test plan is reviewed after each sprint | Test plan is reviewed after complete development. |
| Testing team can take part in the requirements change phase without problems. | It is difficult for the test to initiate any change in needs. |

* **eXtreme Programming (XP)**

Extreme Programming technique is very helpful when there is constantly changing demands or requirements from the customers or when they are not sure about the functionality of the system. It advocates frequent "releases" of the product in short development cycles, which inherently improves the productivity of the system and also introduces a checkpoint where any customer requirements can be easily implemented. The XP develops software keeping customer in the target.

[](https://www.guru99.com/images/11-2014/agile_Processesv1_5.png)

Business requirements are gathered in terms of stories. All those stories are stored in a place called the parking lot.

In this type of methodology, releases are based on the shorter cycles called Iterations with span of 14 days time period. Each iteration includes phases like coding, unit testing and system testing where at each phase some minor or major functionality will be built in the application.

**Extreme programming (XP)** is a [software development methodology](https://en.wikipedia.org/wiki/Software_development_methodology) which is intended to improve [software quality](https://en.wikipedia.org/wiki/Software_quality) and responsiveness to changing customer requirements. As a type of [agile software development](https://en.wikipedia.org/wiki/Agile_software_development), it advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted.

Other elements of extreme programming include: programming [in pairs](https://en.wikipedia.org/wiki/Pair_programming) or doing extensive [code review](https://en.wikipedia.org/wiki/Code_review), [unit testing](https://en.wikipedia.org/wiki/Unit_testing) of all code, [not programming features until they are actually needed](https://en.wikipedia.org/wiki/You_aren%27t_gonna_need_it), a flat management structure, code simplicity and clarity, expecting changes in the customer's requirements as time passes and the problem is better understood, and frequent communication with the customer and among programmers. *The methodology takes its name from the idea that the beneficial elements of traditional software engineering practices are taken to "extreme" levels. As an example,*[*code reviews*](https://en.wikipedia.org/wiki/Code_review)*are considered a beneficial practice; taken to the extreme, code can be reviewed continuously (i.e., the practice of*[*pair programming*](https://en.wikipedia.org/wiki/Pair_programming)*)*.

**Phases of eXtreme programming:**

There are 6 phases available in Agile XP method, and those are explained as follows:

***Planning***

* Identification of stakeholders and sponsors
* Infrastructure Requirements
* [Security](https://www.guru99.com/ethical-hacking-tutorials.html) related information and gathering
* Service Level Agreements and its conditions

***Analysis***

* Capturing of Stories in Parking lot (all items should be either resolved or set as action items.)
* Prioritize stories in Parking lot
* Scrubbing of stories for estimation
* Define Iteration SPAN(Time)
* Resource planning for both Development and QA teams

***Design***

* Break down of tasks
* Test Scenario preparation for each task
* Regression Automation Framework

***Execution***

* Coding
* Unit Testing
* Execution of Manual test scenarios
* Defect Report generation
* Conversion of Manual to Automation regression test cases
* Mid Iteration review
* End of Iteration review

***Wrapping***

* Small Releases
* Regression Testing

(Testing after changes to do retesting of ,impact analysis )

* Demos and reviews
* Develop new stories based on the need
* Process Improvements based on end of iteration review comments

***Closure***

* Pilot Launch
* Training
* Production Launch
* SLA Guarantee assurance
* Review SOA strategy
* Production Support

There are two storyboards available to track the work on a daily basis, and those are listed below for reference.

* Story Cardboard
  + This is a traditional way of collecting all the stories in a board in the form of stick notes to track daily XP activities. As this manual activity involves more effort and time, it is better to switch to an online form.
* Online Storyboard
  + Online tool Storyboard can be used to store the stories. **Several teams can use it** for different purposes.
* **Adaptive Software Development**

Adaptive Software Development practices provide ability to accommodate change and are adaptable in turbulent environments with products evolving with little planning and learning.

* **Phases of ASD Life Cycle**

Adaptive Software Development is cyclical like the Evolutionary model, with the phase names reflecting the unpredictability in the complex systems. The phases in the Adaptive development life cycle are −

* Speculate
* Collaborate
* Learn

These three phases reflect the dynamic nature of Adaptive Software Development. The Adaptive Development explicitly replaces Determinism with Emergence. It goes beyond a mere change in lifecycle to a deeper change in management style. Adaptive Software Development has a dynamic Speculate-Collaborate-Learn Lifecycle.

The Adaptive Software Development Lifecycle focuses on results, not tasks, and the results are identified as application features.



* **Speculate**

The term plan is too deterministic and indicates a reasonably high degree of certainty about the desired result. The implicit and explicit goal of conformance to plan, restricts the manager's ability to steer the project in innovative directions.

In Adaptive Software Development, the term plan is replaced by the term speculate. While speculating, the team does not abandon planning, but it acknowledges the reality of uncertainty in complex problems. Speculate encourages exploration and experimentation. Iterations with short cycles are encouraged.

* **Collaborate**

Complex applications are not built, they evolve. Complex applications require that a large volume of information be collected, analyzed, and applied to the problem. Turbulent environments have high rates of information flow. Hence, complex applications require that a large volume of information be collected, analyzed, and applied to the problem. This results in diverse Knowledge requirements that can only be handled by team collaboration.

Collaborate would require the ability to work jointly to produce results, share knowledge or make decisions.

In the context of project management, Collaboration portrays a balance between managing with traditional management techniques and creating and maintaining the collaborative environment needed for emergence.

* **Learn**

The Learn part of the Lifecycle is vital for the success of the project. Team has to enhance their knowledge constantly, using practices such as −

* Technical Reviews
* Project Retrospectives
* Customer Focus Groups

Reviews should be done after each iteration. Both, the developers and customers examine their assumptions and use the results of each development cycle to learn the direction of the next. The team learns −

* About product changes
* More fundamental changes in underlying assumptions about how the products are being developed

The iterations need to be short, so that the team can learn from small rather than large mistakes.

**Speculate - Collaborate - Learn Cycle as a Whole**

As you observe from the Speculate-Collaborate-Learn cycle, given above, it is obvious that the three phases are nonlinear and overlap.

We observe the following from an Adaptive framework.

* It is difficult to Collaborate without Learning or to Learn without Collaborating.
* It is difficult to Speculate without Learning or to Learn without Speculating.
* It is difficult to Speculate without Collaborating or to Collaborate without Speculating.
* **Dynamic Software Development Method (DSDM)**

DSDM is a Rapid Application Development (RAD) approach to software development and provides an agile project delivery framework. The important aspect of DSDM is that the users are required to be involved actively, and the teams are given the power to make decisions. Frequent delivery of product becomes the active focus with DSDM. The techniques used in DSDM are

1. Time Boxing
2. MoSCoW Rules
3. Prototyping

**(Time Boxing** :- is the approach for completing the project incrementally by breaking it down into splitting the project in portions, each with a fixed budget and a delivery date. For each portion a number of requirements are prioritized and selected. Because time and budget are fixed, the only remaining variables are the requirements. So if a project is running out of time or money the requirements with the lowest priority are omitted)

The DSDM project consists of 7 phases

1. Pre-project
2. Feasibility Study
3. Business Study
4. Functional Model Iteration
5. Design and build Iteration
6. Implementation
7. Post-project

The **MoSCoW method** is a prioritization technique used in management, [business analysis](https://en.wikipedia.org/wiki/Business_analysis), [project management](https://en.wikipedia.org/wiki/Project_management), and [software development](https://en.wikipedia.org/wiki/Software_development) to reach a common understanding with [stakeholders](https://en.wikipedia.org/wiki/Project_stakeholder) on the importance they place on the delivery of each [requirement](https://en.wikipedia.org/wiki/Requirements_analysis); it is also known as *MoSCoW prioritization* or *MoSCoW analysis*.

The term *MoSCoW* itself is an [acronym](https://en.wikipedia.org/wiki/Acronym) derived from the first letter of each of four prioritization categories: **M - *Must have,* S - *Should have,* C - *Could have,* W - *Won't have***

**The interstitial *O*s are added to make the word pronounceable.** While the *O*s are usually in lower-case to indicate that they do not stand for anything, the all-capitals *MOSCOW* is also used.

The **Dynamic Systems Development technique (DSDM)** is an associate degree agile code development approach that provides a framework for building and maintaining systems. The DSDM philosophy is borrowed from a modified version of the sociologist principle—80 % of An application is often delivered in twenty percent of the time it’d desire deliver the entire (100 percent) application.

DSDM is An iterative code method within which every iteration follows the 80% rule that simply enough work is needed for every increment to facilitate movement to the following increment. The remaining detail is often completed later once a lot of business necessities are noted or changes are requested and accommodated.

80% of the project comes from 20% of the system requirements, so as long as those most important 20% of requirements are implemented into the system, the system therefore meets the business needs and that no system is built perfectly in the first try.

The DSDM tool (www.dsdm.org) could be a worldwide cluster of member companies that put together tackle the role of “keeper” of the strategy. The pool has outlined AN [Agile Development Model](https://www.geeksforgeeks.org/software-engineering-agile-development-models/), known as the DSDM life cycle that defines 3 different unvarying cycles, preceded by 2 further life cycle activities:

1. **Feasibility Study:**

It establishes the essential business necessities and constraints related to the applying to be designed then assesses whether or not the application could be a viable candidate for the DSDM method.

1. **Business Study:**

It establishes the use and knowledge necessities that may permit the applying to supply business value; additionally, it is the essential application design and identifies the maintainability necessities for the applying.

1. **Functional Model Iteration:**

It produces a collection of progressive prototypes that demonstrate practicality for the client.

(Note: All DSDM prototypes are supposed to evolve into the deliverable application.) The intent throughout this unvarying cycle is to collect further necessities by eliciting feedback from users as they exercise the paradigm.

1. **Design and Build Iteration:**

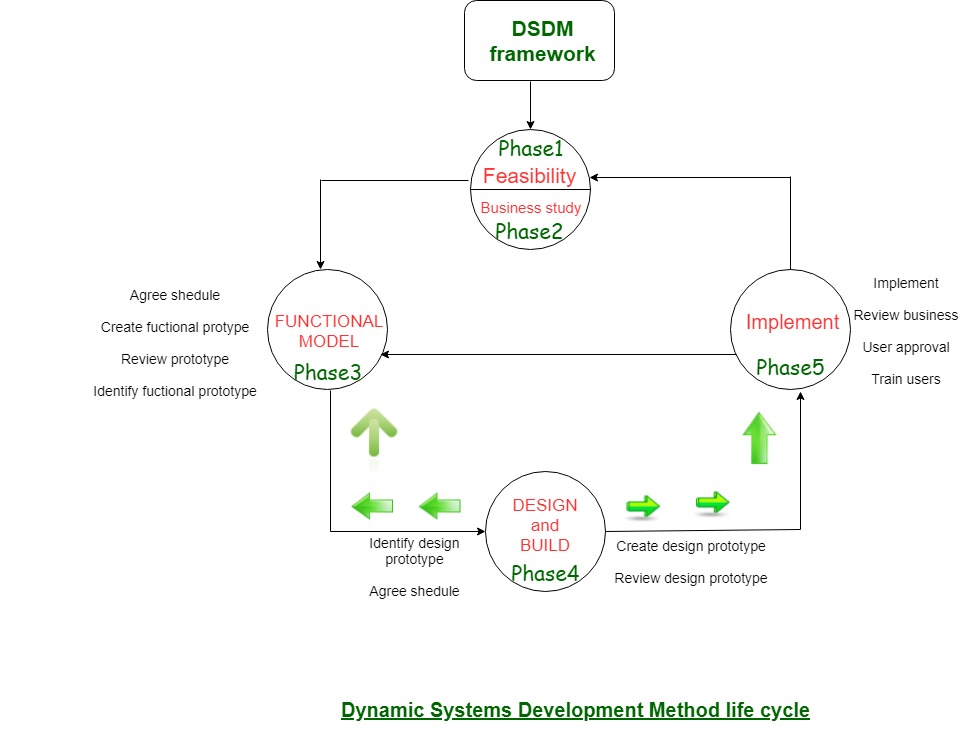
It revisits prototypes designed throughout useful model iteration to make sure that everyone has been designed during a manner that may alter it to supply operational business price for finish users. In some cases, useful model iteration and style and build iteration occur at the same time.

1. **Implementation:**  
   It places the newest code increment (an “operationalized” prototype) into the operational surroundings. It ought to be noted that:
   * **(a)** the increment might not 100% complete or,
   * **(b)** changes are also requested because the increment is placed into place. In either case, DSDM development work continues by returning to the useful model iteration activity.

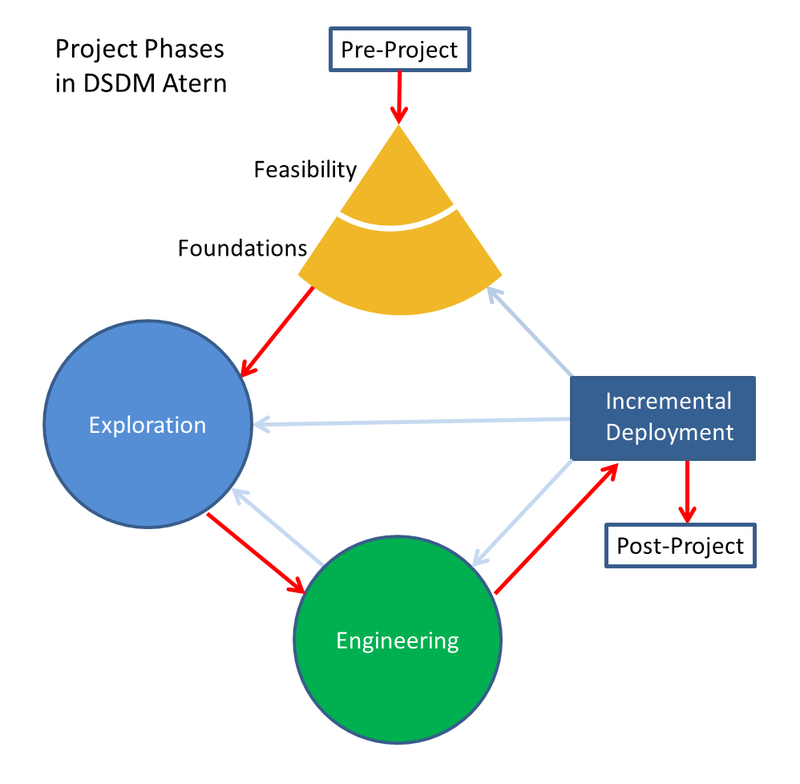
**There are eight principles** underpinning DSDM . These principles direct the team in the attitude they must take and the mindset they must adopt to deliver consistently.

1. Focus on the business need
2. Deliver on time
3. Collaborate
4. Never compromise quality
5. Build incrementally from firm foundations
6. Develop iteratively
7. Communicate continuously and clearly
8. Demonstrate control

Below diagram describe the DSDM life cycle:



DSDM is often combined with XP to supply a mixed approach that defines a solid method model (the DSDM life cycle) with the barmy and bolt practices (XP) that are needed to create code increments. additionally, the ASD ideas of collaboration and self-organizing groups are often tailored to a combined method model



* **Scrum**

**Scrum is a framework** that helps agile teams to work together. Using it, the team members can deliver and sustain the complex product. It encourages the team to learn through practice, self-organize while working on the problem. Scrum is a work done through the framework and continuously shipping values to customers.

It is the most frequent software that is used by the development team. Its principle and lessons can be applied to all kinds of teamwork. Its policy and experiences is a reason of popularity of Scrum framework. The Scrum describes a set of tools, meetings, and roles that help the teams structure. It also manages the work done by the team

Scrum is Agile Development Framework for managing product development. It is flexible and holistic product development approach. In this methodology development and QA teams work as one unit.

The term is borrowed from [rugby](https://en.wikipedia.org/wiki/Rugby_football), where a [scrum](https://en.wikipedia.org/wiki/Scrum_(rugby)) is a formation of players. The term *scrum* was chosen by the paper's authors because it emphasizes teamwork



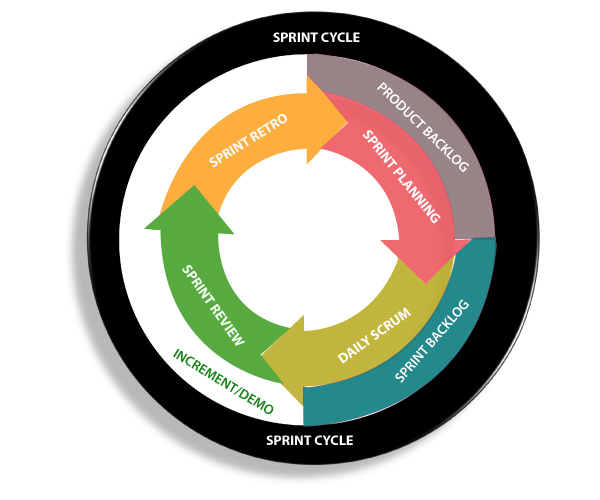
**The framework**

Scrum and agile are not the same thing because Scrum focused on continuous improvement, which is a core foundation of agile. Scrum framework focuses on ongoing getting work done.

**What are sprints?**

With scrum, a product is built in a series of repetition called sprints. It breaks down big complex projects into bite-size pieces. It makes projects more manageable, allows teams to ship high quality, work faster, and more frequently. The sprints give them more flexibility to adapt to the changes.

Sprints are a short, time-boxed period for Scrum team that works to complete a set amount of work. Sprints are the core component of Scrum and agile methodology. The right sprints will help our agile team to ship better software.



**What is sprint plan?**

Sprint plan is an action in Scrum that kicks off the sprint. The primary purpose of sprint plan is to define what can deliver in the sprint. It also focuses on how the work will be achieved. It is done in combination with the whole Scrum team members.

The sprint is a set of the period where all the work to be done. Before we start the development, we have to set up the sprint. We need to describe how long time is required to achieve the sprint goal and where we are going to start.

**Factors affecting Sprint planning**

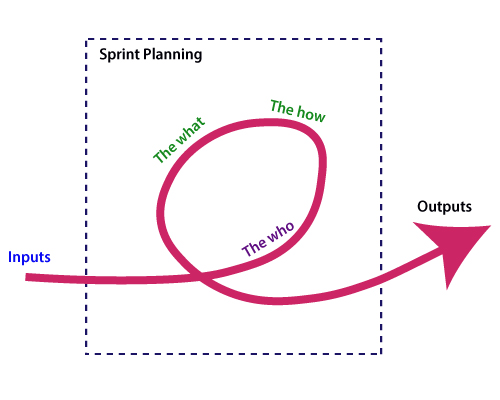
**The What:** The product owner describes the goal of the sprint and the backlog items which contribute to achieve that goal.

**The How:** Agile development team plans its necessary work on how to achieve and deliver the sprint goal.

**The Who:** The product owner defines the goal based on the value that the customers seek. And the developer needs to understand how they can or cannot deliver that goal.

**The Inputs:** The product backlog provides the list of input stuff that could potentially be part of the current sprint. The team looks over the existing work done in incremental ways.

**The Outputs:** The critical outcome of sprint planning is to meet described team goal. The product set the goal of sprint and how they will start working towards the goal.



**What is the product backlog?**

A product backlog is a registered list of work for the development team. It is driven from the roadmap and its requirements. The essential task is represented at the top of the product backlog so that the team member knows what to deliver first. The developer team doesn't work through the backlog from the product owner's side and product owner doesn't push the work to the developer team. The developer team pulls work from the product backlog.

**Backlog starts with the two "R"s**

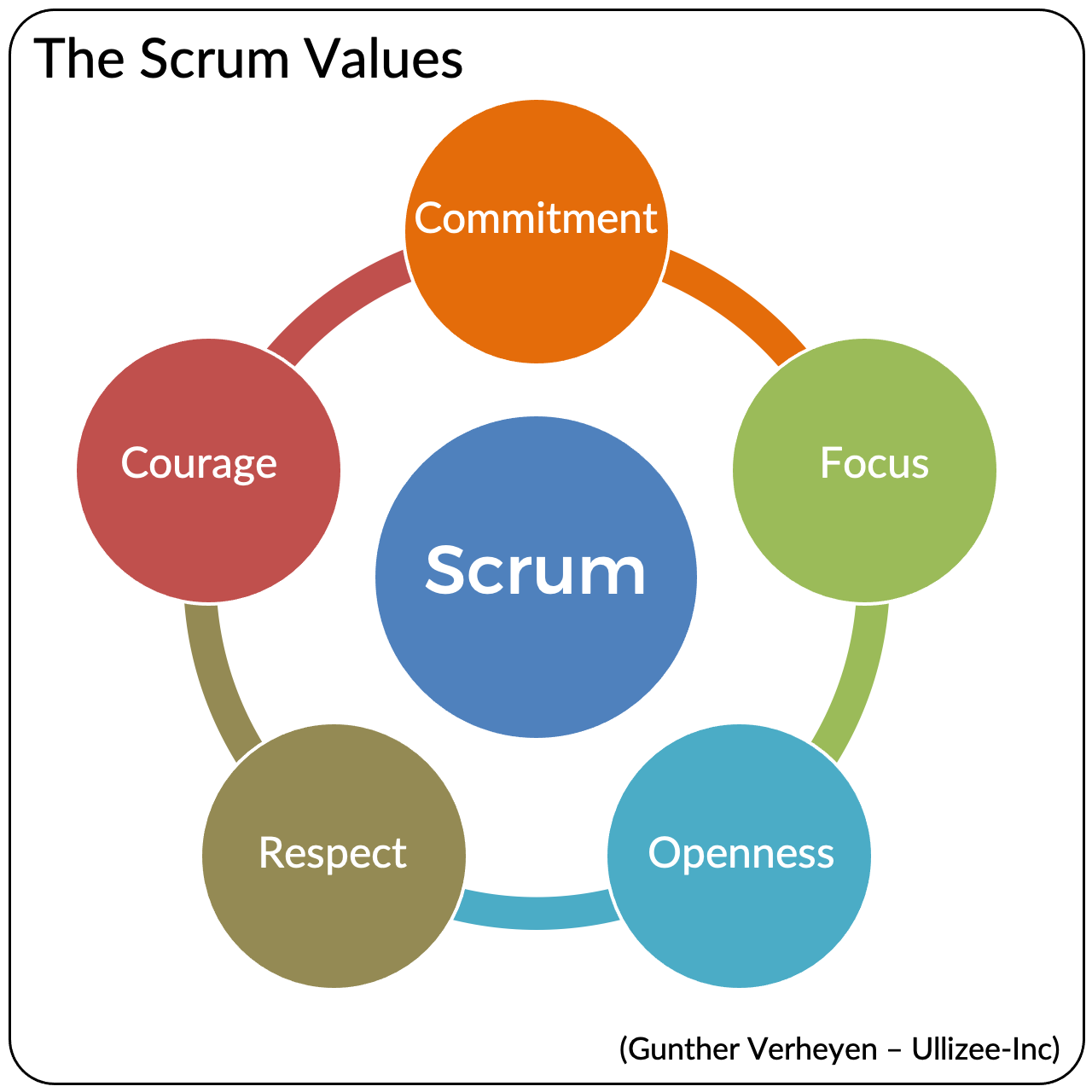
The fundamental product backlog is provided by a

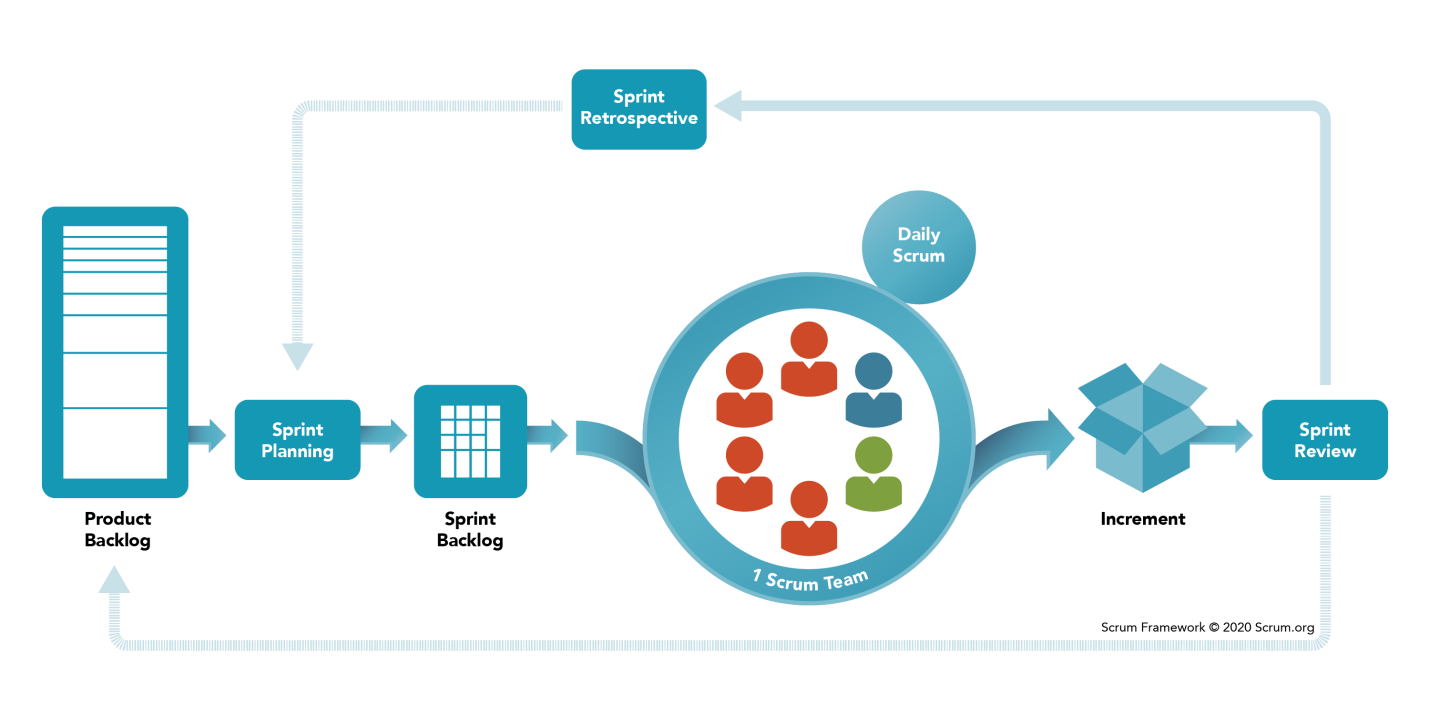
team's **roadmap** and **requirements**. Roadmap repetition breaks down into several epics, and each epic will have several requirements and user stories.

The product owner organizes each of the customer stories into a single list. This story is organized for the development team. The product owner chooses to deliver first complete epic.

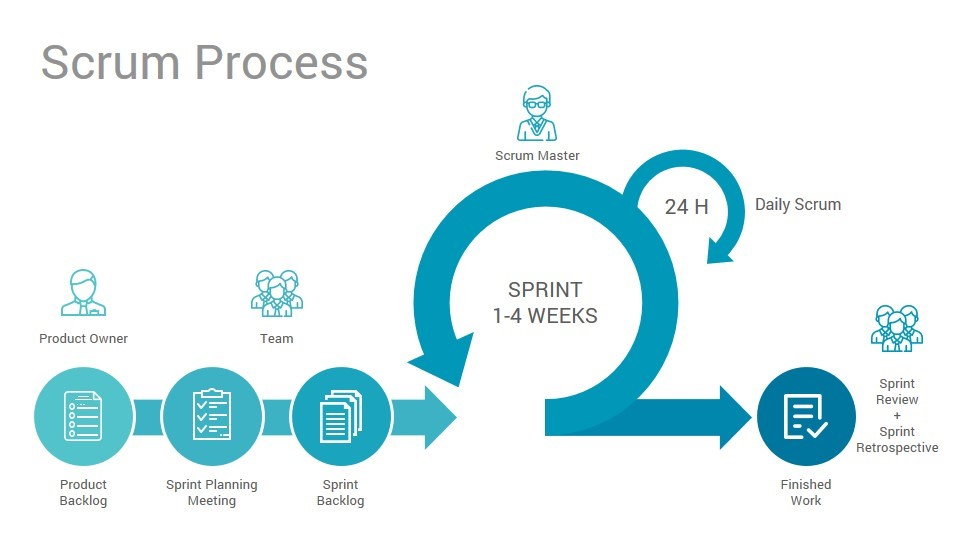
**The factors that influence a product owner's prioritization**

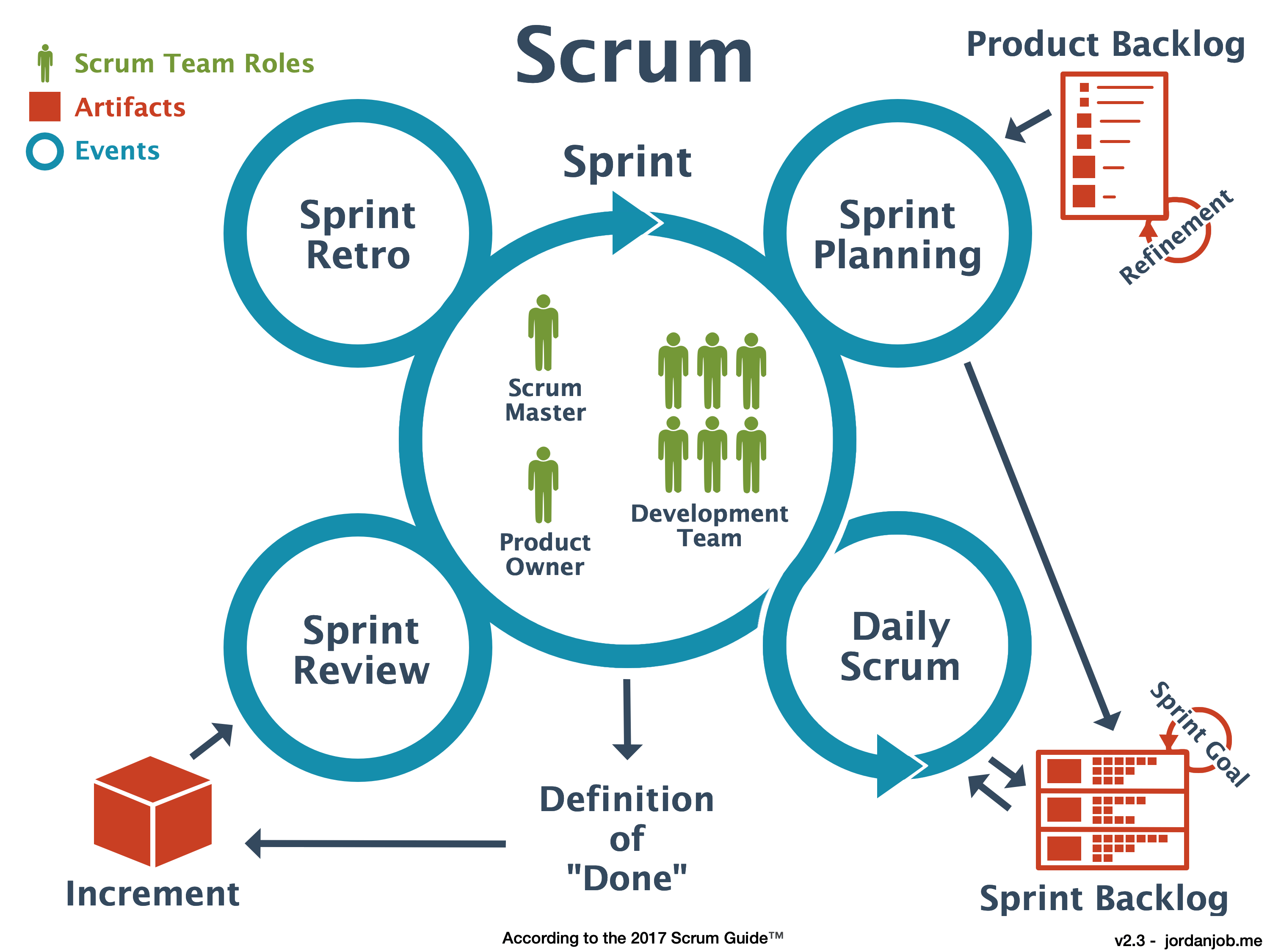
* Priority of customer
* Importance of getting feedback
* Relative implementation difficulty
* Symbiotic relationships between work items

****

****

**Scrum Framework**

****

****

**Product owner:-**

The product owner, representing the product's [stakeholders](https://en.wikipedia.org/wiki/Stakeholder_(corporate)) and the [voice of the customer](https://en.wikipedia.org/wiki/Voice_of_the_customer) (or may represent the desires of a committee), is responsible for delivering good business results. Hence, the product owner is accountable for the product backlog and for maximizing the value that the team delivers. The product owner defines the product in terms of customer-centric outcomes (typically - but not limited to - [user stories](https://en.wikipedia.org/wiki/User_story)), adds them to the product backlog, and [prioritizes](https://en.wikipedia.org/wiki/Requirement_prioritization) them based on importance and dependencies. A scrum team should have only one product owner (although a product owner could support more than one team) and it is strongly advised against combining this role with the role of the scrum master. The product owner should focus on the business side of product development and spend the majority of time liaising with stakeholders and the team. The product owner does not dictate how the team reaches a technical solution, but seeks consensus among team members. This role is crucial and requires a deep understanding of both sides: the business and the engineers (developers) in the scrum team. Therefore, a good product owner should be able to communicate what the business needs, ask why they need it (because there may be better ways to achieve that), and convey the message to all stakeholders including the developers using technical language, as required. The product owner uses Scrum’s empirical tools to manage highly complex work while controlling risk and achieving value.

Communication is a core responsibility of the product owner. The ability to convey priorities and empathize with team members and stakeholders is vital to steer product development in the right direction. The product owner role bridges the communication gap between the team and its stakeholders, serving as a proxy for stakeholders to the team and as a team representative to the overall stakeholder community.

As the face of the team to the stakeholders, the following are some of the communication tasks of the product owner to the stakeholders:

* Define and announce releases.
* Communicate delivery and product status.
* Share progress during governance meetings.
* Share significant RIDAs (risks, impediments/obstacle, dependencies, and assumptions) with stakeholders.
* Negotiate priorities, scope, funding, and schedule.
* Ensure that the product backlog is visible, transparent and clear.

Empathy is a key attribute for a product owner to have—the ability to put one's self in another's shoes. A product owner converses with different stakeholders with a variety of backgrounds, job roles, and objectives - and should be able to appreciate these different points of view. To be effective, it is wise for a product owner to know the level of detail the audience needs. The developers need thorough feedback and specifications so they can build a product up to expectation, while an executive sponsor may just need summaries of progress. Providing more information than necessary may lose stakeholder interest and waste time. A direct means of communication is preferred by seasoned product owners.

A product owner's ability to communicate effectively is also enhanced by being skilled in techniques that identify stakeholder needs, negotiate priorities between stakeholder interests, and collaborate with developers to ensure effective implementation of requirements.

**Sprint :-**

A sprint (also known as *iteration* or *timebox*) is the basic unit of development in Scrum. The sprint is a [timeboxed](https://en.wikipedia.org/wiki/Timeboxing) effort; that is, the length is agreed and fixed in advance for each sprint and is normally between one week and one month, with two weeks being the most common.

Each sprint starts with a *sprint planning* event in which a sprint goal is crafted and a sprint backlog emerges, containing intended work for the upcoming sprint. Each sprint ends with two events:

* a *sprint review* (progress shown to stakeholders)
* a *sprint retrospective* (identify lessons and improvements for the next sprints).

Scrum emphasizes valuable, useful output at the end of the sprint that is really done. In the case of software, this likely includes that products are fully integrated, tested and documented, and potentially releasable.

**Sprint planning**

At the beginning of a sprint, the scrum team holds a sprint planning event to:

* Agree the sprint goal, a short description of what they forecast to deliver by sprint end, based on the priorities set by the product owner
* Select product backlog items that contribute towards this goal
* Form a sprint backlog by mutually discussing and agreeing on which items are intended to be done during that sprint

The maximum duration of sprint planning is eight hours for a 4 week sprint.As the detailed work is elaborated, some product backlog items may be split or returned to the product backlog if the team believes they cannot complete that work in a single sprint

**Daily Scrum**

Each day during a sprint, the developers hold a daily scrum (sometimes conducted [standing up](https://en.wikipedia.org/wiki/Stand-up_meeting)) with specific guidelines:

All developers come prepared. The daily scrum:

* is focused on inspecting progress towards the sprint goal
* should happen at the same time and place every day
* is limited ([timeboxed](https://en.wikipedia.org/wiki/Timeboxing)) to fifteen minutes
* is conducted however the team decides
* may include others, though only developers should speak.
* may be facilitated by the Scrum Master
* may identify impediments to progress (e.g., stumbling block, risk, issue, delayed dependency, assumption proved unfounded)
* does not feature discussions
* is NOT a "status report" or a means of updating progress charts

No detailed discussions should happen during the daily scrum. Once over, individual members can discuss issues in detail, often known as a 'breakout session' or an 'after party'. Blockers identified should be collectively discussed outside of the daily scrum with a view to working toward a resolution.

Where the team does not see the value in this event, it is the responsibility of the scrum master to determine why and educate the team and stakeholders about Scrum principles, or encourage the team to design their own method of keeping the team fully informed of sprint progress.

**Sprint review**

Conducted at the end of a sprint, the team:

* presents the completed work to the stakeholders (a.k.a. the [demo](https://en.wikipedia.org/wiki/Technology_demo))
* collaborates with stakeholders on topics such as:
  + inviting feedback about the completed product increment
  + discussing the impact of incomplete work (planned or otherwise)
  + receiving suggestions for upcoming work (guidance of what to work on next)

Product Owners should see this event as a valuable opportunity to review and refine the product backlog with stakeholders.

**Guidelines for sprint reviews:**

* Incomplete work should not be demonstrated; although stakeholders should be presented with product increments they will be receiving, but can also request to see work in progress if necessary. However, the team should only be prepared to show what has been done.
* The recommended duration is two hours for a two-week sprint (proportional for other sprint-durations).

**Sprint retrospective**

At the sprint retrospective, the team:

* reflects on the past sprint
* identifies and agrees on [continuous process improvement](https://en.wikipedia.org/wiki/Continual_improvement_process) actions

**Guidelines for sprint retrospectives:**

* Three suggested areas to consider in sprint retrospectives are:
  + What went well during the sprint?
  + What did not go well?
  + What could we do differently the next sprint?
* The recommended duration is one-and-a-half hours for a two-week sprint (proportional for other sprint duration(s)).

The scrum master may facilitate this event, but their presence is not considered mandatory.

**What is Scrum Master?**

Scrum master is a person who helps other people to understand Scrum and serves the project team by removing obstacles. He also helps in simplifying project complexities.

The Scrum master also needs to make sure that development team works based on the core values of Scrum. He is often considered a coach for the team, helping the team do the best work they possibly can. Moreover, he ensures that the Scrum adoption is successful in the enterprise.

Scrum master acts as a heart of Scrum project. He needs to perform responsibilities like:

* Scrum Master needs to assure that team meets its business objectives
* Scrum Master fosters collaborative environment in the team
* The Scrum master does the planning, team backlog grooming, sprint demo, sprint retrospective.
* Guides Scrum processes and helps to maintain integrity of Scrum values
* Promote improved engineering practices like TDD, automated testing and continuous integration.
* Making sure that every stakeholder should be present at the meeting at the given time
* Works together with other team members to ensure dependencies and risk are distributed across Scrum teams.
* Conducting feasibility studies, writing & validating specifications.

**What is Scrum master is not?**

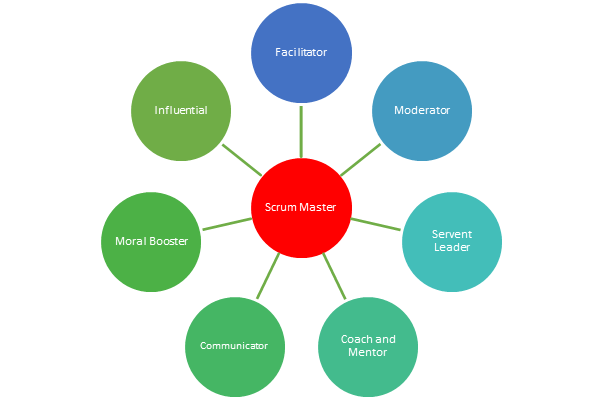
The Scrum master is not a project manager. Both roles are different in the agile process. He will not guide a team in every phase of the development. He is neither responsible for any technical decision nor for managing the business with clients. He will never lead the team or make decisions on behalf of the team.

**Scrum Master Skills**

The Scrum master is a very important person in the agile Scrum process. He should have multiple skills including technical, scrum process and soft skills.

Scrum master must know different techniques and practices to manage self-organizing teams. He should show honesty, integrity, trust, and respect for the team. He should have expertise in Agile, IT coaching, and presentation. Moreover, Scrum master should able to schedules meetings in such a way that it doesn't affect the regular work of the team members.

**Characteristic of the Good Scrum master**

[](https://www.guru99.com/images/1/120817_0726_ScrumMaster1.png)

**Characteristic of Scrum Master**

Following are important characteristics of Scrum master:

**Servant-Leader:** Servant leadership is a social leadership style. Scrum master act as a servant leader to attain project objectives through service to others.

**Morale booster:** The Scrum master also acts as a Morale booster by encouraging the team to do their work more effectively.

**Facilitator**: Scrum master plays a role of facilitator as he helps the team to understand their common project-related objective. He also assists the team to achieve these objectives while remaining neutral.

**Coach:** Scrum masters coaches the team to become self-organized and also guides them on the path of continuous improvement.

**Agent:** Scrum master also plays a role of an agent which helps to make the necessary changes allowing a better implementation of Scrum.

**Mentor:** The Scrum Master should share his knowledge, experiences, and even his opinions just like a mentor.

**Remover of obstacles:** Scrum master should remove all the obstacles preventing the development team to work in the best conditions.

**Manager of the framework:** The Scrum Master is also responsible for the framework within which the team works.

**Observer:** He needs to constantly observe everything even the smallest activities of the team members.

**Daily Activities of Scrum master:**

**Activity 1:** Capacity and velocity planning using spreadsheet and Jira.

**Activity 2:** Determine implementable stories available with points remaining from the sprint review.

**Activity 3:** Review previous sprint velocity along with the team.

**Activity 4:**Estimating teams' capacity

**Activity 5:**Calculating number of points completed by team

**Activity 6:** The team member provides all known tasks and estimates task effort within an hour. Scrum Master should ensure task given to members are never above their capacity.

**Activity 7:** Scrum master also needs to ensure that the tasks in never more than 16 hours. If at all it is bigger then that he should tell the team to make it in the break them down.

The above is just a small sample of the many activities a Scrum Master is involved in.

**Best practices for Scrum master**

* Scrum master should listen to all the queries of developers, project manager, and owner to understand their issues.
* He should entrust his unwavering faith in his team members that they will complete the allocated work.
* Scrum master should avoid doing multitasking in his/her work
* He should give respect even to the smallest team member and listen to everyone's perspective.
* Show genuine appreciation to the team for their successful accomplishments

**Why is Scrum master's role is difficult?**

First of all, management of large size self-organizing teams is not an easy task. It needs lots of willpower. Scrum master often needs to manage conflicts between team members and project manager.

He may head the entire Scrum management project still he remains without any power. Scrum master also needs to learn about motivation, communication, behavioral economics and a whole host of other social science which surely needs lots of time outside of his job.

**Difference between Agile and Scrum (Agile vs Scrum)**

[Agile](https://www.javatpoint.com/agile-methodology) is an iterative approach of software development methodology using short iterations of 1 to 4 weeks. Due to the agile methodology, the development process is aligned to deliver the changing business requirement. Using Agile methodology, the software is distributed with faster and fewer changes.

[Scrum](https://www.javatpoint.com/agile-scrum) is a framework of agile that helps agile teams to work together. Using it, the team members development, deliver and sustain the complex product. It encourages the team to learn through practice, self-organize while working on the problem. Scum is a work done through the framework and continuously shipping values to customers.

**Agile**

1. Agile is an **iterative and incremental approach** to software development methodology.
2. In this approach, the **leadership** plays an important role.
3. Agile software development is highly suitable for the **medium or large project**.
4. **Flexibility** is the most significant advantage of agile as it quickly reacts to changes.
5. Agile involves **face-to-face communication** and collaboration between the members of various cross-functional teams.
6. Agile development needs **frequent delivery** to the end user for their feedback.
7. In this development, each step like requirements, analysis, design, are **continually monitored** during the lifecycle.
8. The **project leader** takes cares of all the tasks in the agile method.
9. End-user may give their **feedback during the development** process. So, the end product will be more useful.
10. **Delivery and update** of the software are taking place regularly.
11. Design and execution should be kept **simple**.
12. The priority of agile development is always to satisfy the customer by providing **continuous delivery** of valuable software.
13. Working software is the most **fundamental measure** of progress.
14. It is best to have **face-to-face communication** to get as close to the project goal as possible.

**Scrum**

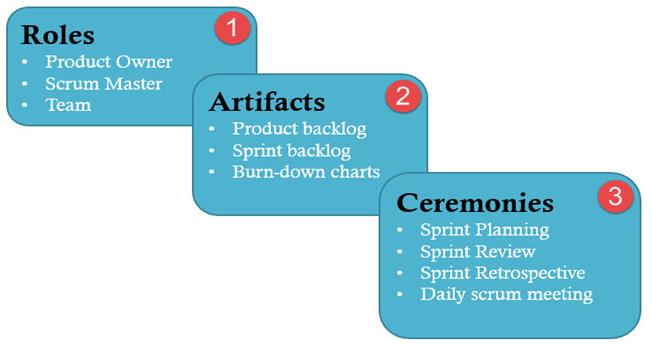
1. Scrum is a framework of agile methodology. In which **incremental builds** are delivered to end user in every two to three weeks.
2. Scrum's team is **self-organized**, cross-functional team.
3. Scrum is used in the project where the requirement **rapidly** changes. v
4. A compared to agile it is more **rigid**. So that there are no chances of frequent change.
5. In **daily stand up meeting** the teamwork is achieved with a fixed role assigned to team members, scrum master, and product owner.
6. **No need to change many more** while implementing scrum process.
7. In this process, a **build is delivered** after each sprint to the client for their feedback.
8. After every sprint a demonstration of functionality is provided. So that the **regular feedback** can be taken before next sprint.
9. There is no team leader, so the **entire team handles the issues** or problems.
10. When the team completes the **current sprint activity**, then the next sprint is planned.
11. Design and execution can be **innovative and experimental**.
12. The **daily sprint meeting** is organized to review the feedback to decide the future progress of the project.
13. Working software is **not a fundamental measure**.
14. The target of the Scrum team is to deliver **maximum business value**.

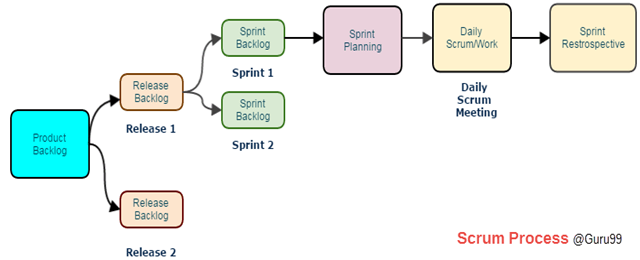
**Key Features of Scrum Methodology**

Following are Key Features of Scrum-

* Scrum has a short fixed schedule of release cycles with adjustable scope known as **sprints** to address rapidly changing development needs. Each release could have multiple sprints. Each Scrum Project could have multiple Release Cycles.
* A repeating sequence of **meetings, events, and milestones**
* A practice of testing and implementing new requirements, known as **stories**, to make sure some work is released ready after each sprint

Scrum is based on the following 3 Pillars-

[](https://www.guru99.com/images/11-2014/112714_1232_ScrumTestin1.jpg)

[](https://www.guru99.com/images/11-2014/112714_1232_ScrumTestin2.png)

A scrum process includes

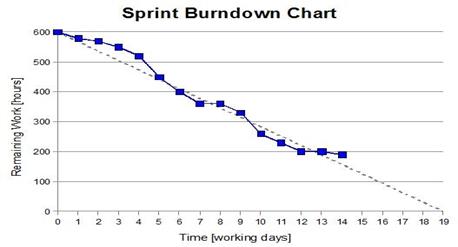
* **User stories:**They are a short explanation of functionalities of the system under test. Example for Insurance Provider is – "Premium can be paid using the online system."
* **Product Backlog:** It is a collection of user stories captured for a scrum product.**The product owner prepares** and maintains the product backlog. It is prioritized by the product owner, and anyone can add to it with approval from the product owner.
* **Release Backlog:**A release is a time frame in which the number of iterations is completed.**The product owner co-ordinates** with the scrum master to decide which stories should be targeted for a release. Stories in the release backlog are targeted to be completed in a release.
* **Sprints:**It is a set period of time to complete the user stories, decided by the product owner and developer team, usually 2-4 weeks of time.
* **Sprint Backlog:**It's a set of user stories to be completed in a sprint. During sprint backlog, work is never assigned, and the team signs up for work on their own. It is owned and managed by the team while the estimated work remaining is updated daily. It is the list of task that has to be performed in Sprint
* **Block List:**It is a list of blocks and unmade decisions owned by scrum master and updated daily
* **Burndown chart:**Burn-down chart represents overall progress of the work in progress and work completed throughout the process. It represents in a graph format the stories and features not completed

**Ceremonies (Processes) in Scrum**

* **Sprint Planning:**A sprint begins with the team importing stories from the release backlog into the sprint backlog; it is hosted by scrum master. The Testers estimate effort to test the various stories in the Sprint Backlog.
* **Daily Scrum:**It is hosted by scrum master, it last about 15 minutes. During Daily Scrum, the members will discuss the work completed the previous day, the planned work for the next day and issues faced during a sprint. During daily stand-up meeting team progress is tracked.
* **Sprint Review/ Retrospective:**It is also hosted by scrum master, it last about 2-4 hours and discuss what the team has accomplished in the last sprint and what lessons were learned

**Burn down chart:** Each day, Scrum Master records the estimated remaining work for the sprint. This is nothing but the Burn Down Chart. It is updated daily.

A burndown chart gives a quick overview of the project progress, this chart contains information like the total amount of work in the project that must be completed, amount of work completed during each sprint and so on.

* [](https://www.guru99.com/images/11-2014/112714_1232_ScrumTestin4.jpg)
* **Crystal Agile Process** :-

*Crystal It is a subset of Agile Methodology because it use iteration and incremental development same as Agile Model*

**What is the History of the Crystal Method?**

[Alistair Cockburn](https://alistair.cockburn.us/), credited as one of the original popularizers of [agile](https://www.productplan.com/agile-product-management/), developed the Crystal method for IBM in 1991. He decided to focus not on developing specific step-by-step development strategies that would work across the board for teams involved in any project, but instead to develop guidelines for team collaboration and communication. The traits of Cockburn’s Crystal method were therefore all based around the team itself:

* Human-powered (meaning the project should be flexible and tailored to the needs and the preferred work modalities of people involved)
* Adaptive (meaning the approach uses no fixed tools but can be altered to meet the team’s specific needs)
* Ultra-light (meaning this methodology does not require much documentation or reporting)

**What is the Crystal Method?**

Crystal is an [agile framework](https://www.productplan.com/glossary/agile-framework/) focusing on individuals and their interactions, as opposed to processes and tools. In other words, this framework is a direct outgrowth of one of the core values articulated in the [Agile Manifesto](https://www.agilealliance.org/agile101/the-agile-manifesto/).

The Crystal agile framework is built on two core beliefs:

* Teams can find ways on their own to improve and optimize their workflows
* Every project is unique and always changing, which is why that project’s team is best suited to determine how it will tackle the work

Crystal is an agile methodology for software development. It places focus on people over processes, to empower teams to find their own solutions for each project rather than being constricted with rigid methodologies

These are categorized by color, according to the number of people in the project;

* Crystal clear - Teams with less than 8 people
* Crystal yellow - Teams with between 10 and 20 people
* Crystal orange - Teams with between 20-50 people
* Crystal red - Teams with between 50-100 people

**Crystal Clear** :- Support Fixed price contract ,Require Documentation, focus on people not process or artifacts

**Crystal Yellow :-** Easy Communication, clear ownership of code area, Feedback, mission statement , Monthly improvement.

**Crystal Orange** :- Incremental Development , New release after 3-4 months , Each release is called “Increment “ , Designed for medium size project.

**Crystal Red:**- Continuous Learning process , Maximum process, Minimum distraction , Maximum defect free ,frequent delivery and communication

**Crystal Method focus on :-**

* People
* Interaction
* Community
* Skills
* Talent
* Communications

**Key principles of the crystal**[**agile**](https://airfocus.com/glossary/what-is-agile/)**framework**

At the heart of the crystal, a family is seven principles. The first three are compulsory for all crystal approaches, but the rest are optional and can be adopted if appropriate:

**#1: Frequent delivery**

You should deliver code regularly to your real users. Without this, you might be building a product nobody needs.

**#2: Reflective improvement**

Look back on what you've done, how you've done it, and why. As a team, reflect and decide how to improve it in the future.

**#3: Osmotic communication**

Cockburn believed that co-location (having teams in the same physical space) is critical as it allows information to flow between team members, as if by osmosis.

**#4: Personal safety**

Team members should feel safe to discuss ideas openly, without fear of ridicule. There are no wrong answers or bad suggestions in a crystal team.

**#5: Focus on work**

Team members should know what to work on next and be able to do it. This requires clear communication and documentation when required.

**#6: Access to subject matter experts and users**

Team members should be able to get feedback from real users and experts when required.

**#7: Technical tooling**

Even back in the 1990s, Cockburn said development teams should have access to tooling’s like continuous deployment, automated testing, and configuration management. This means errors and mistakes can be caught quickly without human intervention.

Crystal Methods, which is a collection of Agile software development approaches, focuses primarily on people and the interaction among them while they work on a software development project. There is also a focus on business-criticality and business-priority of the system under development. Unlike traditional development methods, Crystal doesn’t fix the tools and techniques of development, but keeps people and processes at the core of the development process. However, it is not only the people or the processes that are important, rather the interaction between the two that is most important.

* **Crystal is “human-powered”**—This means that people are the most important aspect of Crystal, and all the processes and tools are relative to them. Crystal believes that software development is essentially a human activity, so people involved in this activity are vital while the processes should be modelled to meet the requirements of the team, not the other way around. Crystal emphasizes that development teams are self-sufficient and self-organizing, so they are capable of streamlining the processes as the development process progresses and become more organized and competent.
* **Crystal is “adaptive”—**First of all, it should be remembered that Crystal is not a set of prescribed tools and techniques for software development; rather, it is an approach. So, the processes and tools are not fixed, but have to be adjusted to the requirements  and characteristics of the project. In other words, Crystal is a “stretch-to-fit” methodology, because each project is unique and require methods that suit the business requirements and that satisfy the technical requirements of the project.
* **Crystal is “ultra light”**—Crystal is known as a “lightweight methodology.” This is because Crystal doesn’t advocate too much documentation, overhead management and reporting. Instead, it believes in keeping things light and focusing on developing business-valued and functional software. For this, teams following the Crystal approach work toward enhancing free and open communication among  team members as well as establishing transparent flow of information between developers and stakeholders.

**How does Crystal operate?**

Crystal is not a set of prescribed development tools and methods, but a family of various development approaches. At the beginning of the project, the processes and tools are not fixed but are decided by considering the business requirements and technical needs of the project. When deciding whether Crystal is the right methodology for a project, consider comfort, discretionary money, essential money and life along with the size of the team working on a particular project. Various methodologies in the Crystal family are known as the various “weights” of the Crystal approach and are represented by different colors of the spectrum.

**What are the main practices recommended by Crystal?**

Crystal is precise about certain practices because these are crucial for the successful implementation of the Crystal approach onto any project. These practices include:

* **An iterative and incremental development approach**—The project is developed in iterations that are generally time boxed. The feature delivered at the end of an iteration is integrated into the overall system. User feedback taken at the end of an iteration is used to plan the next iteration; and, new and additional features are added in every subsequent iteration. All this results in refinement and completion of the overall software.
* **Active user involvement—**This is a must because Crystal is a people-centric approach and emphasizes transparency. So, users are not only actively involved but also regularly informed about the progress of the project.
* **Delivering on commitments**—The team endeavors to ensure frequent delivery of client-valued, potentially-shippable functionalities. It is to this end that Crystal follows an iterative and an incremental development approach.

**Advantages of using the crystal agile framework:-**

* Teams have a lot of autonomy to work in the way they deem most effective
* Teams communicate directly with each other, reducing management overhead
* The framework can adapt as a team grows or shrinks
* The adaptive approach lets teams respond well to changing requirements

**Disadvantages of using the crystal agile framework :-**

* Lack of structure can slow down inexperienced teams
* Not clear on how remote teams can share knowledge informally
* Lack of rigid planning can lead to confusion and loss of focus
* **Feature Driven Development ( FDD) Agile Methodology :-**

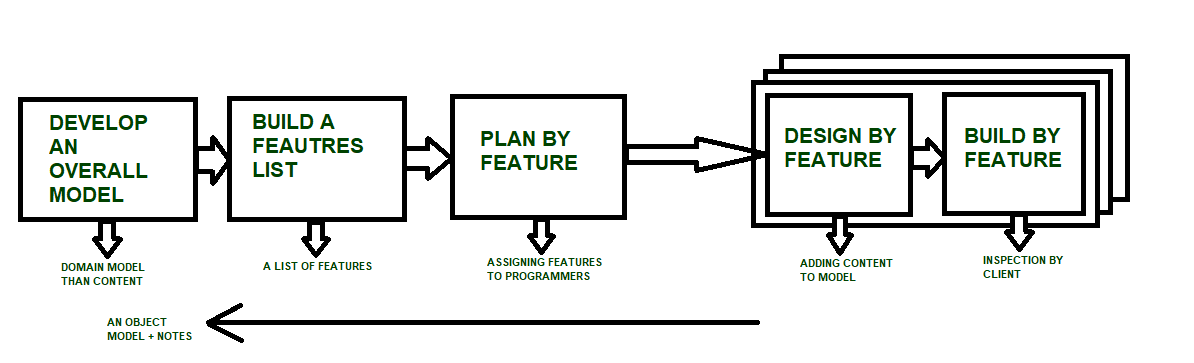
An Agile methodology for developing software, Feature-Driven Development (FDD) is customer-centric, iterative, and incremental, with the goal of delivering tangible software results often and efficiently. FDD in Agile encourages status reporting at all levels, which helps to track progress and results.

FDD allows teams to update the project regularly and identify errors quickly. Plus, clients can be provided with information and substantial results at any time. FDD is a favorite method among development teams because it helps reduce two known morale-killers in the development world: Confusion and rework.

First applied in 1997 during [a project for a Singapore bank](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Feature-driven_development.html), FDD was developed and refined by Jeff De Luca, Peter Coad and others. The original project took 15 months with 50 people, and it worked; it was followed by a second, 18-month long, 250-person project.

**FDD Lifecycle**

* Build overall model
* Build feature list
* Plan by feature
* Design by feature
* Build by feature



**Characteristics of FDD**

* **Short iterative:** FDD lifecycle works in simple and short iterations to efficiently finish the work on time and gives good pace for large projects.
* **Customer focused:** This agile practice is totally based on inspection of each feature by client and then pushed to main build code.
* **Structured and feature focused:** Initial activities in lifecycle builds the domain model and features list in the beginning of timeline and more than 70% of efforts are given to last 2 activities.
* **Frequent releases:** Feature-driven development provides continuous releases of features in the software and retaining continuous success of the project.

**Advantages of FDD**

* Reporting at all levels leads to easier progress tracking.
* FDD provides continuous success for larger size of teams and projects.
* Reduction in risks is observed as whole model and design is build in smaller segments.
* FDD provides greater accuracy in cost estimation of the project due to feature segmentation.

**Disadvantages of FDD**

* This agile practice is not good for smaller projects.
* There is high dependency on lead programmers, designers and mentors.
* There is lack of documentation which can create an issue afterwards.