CSE 4513

Lec – 2~3

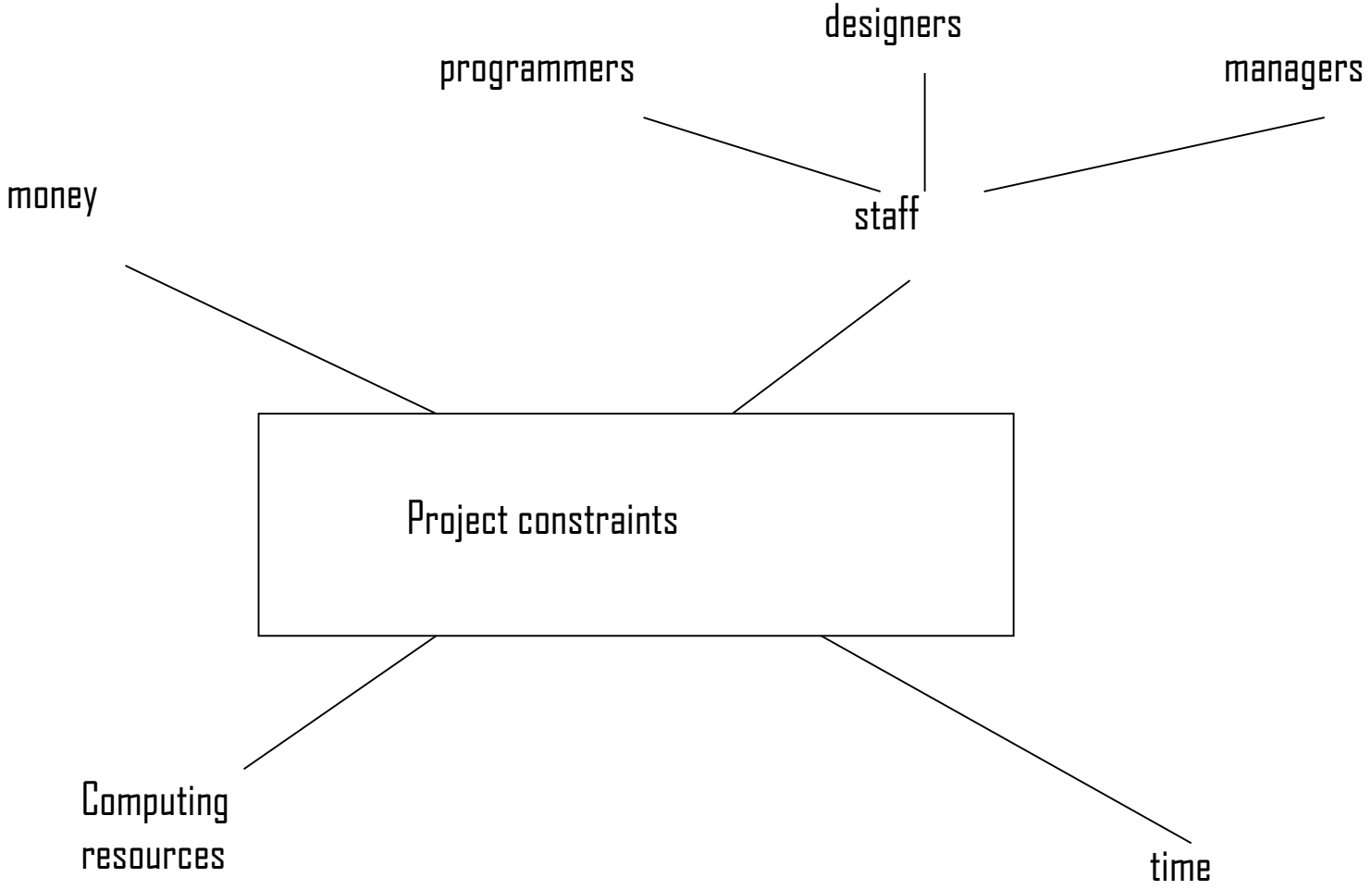
SW Development Life Cycle (SDLC) Models

Inherent Problems with Software Development

- ✓ **Requirements are complex**
  - ➢ The client does not know the functional requirements in advance
- ✓ **Requirements may be changing**
  - ➢ Technology enablers introduce new possibilities to deal with nonfunctional requirements
- ✓ **Frequent changes are difficult to manage**
  - ➢ Identifying milestones and cost estimation is difficult
- ✓ **There is more than one software system**
  - ➢ New system must be backward compatible with existing system (“legacy system”)
  - ➢ Phased development: Need to distinguish between the system under development and already

All these lead us to software life cycle modeling

CONSTRAINTS OF SOFTWARE DEVELOPMENT PROJECT



Examples of SW Project Constraints

SW Development Life Cycle (SDLC) Models

Popular Software developing life cycles (SDLC) models:

- ✓ Waterfall Model
- ✓ V-Shaped Model
- ✓ Iterative Incremental Model
- ✓ Spiral Model
- ✓ Prototyping Model
- ✓ Agile development

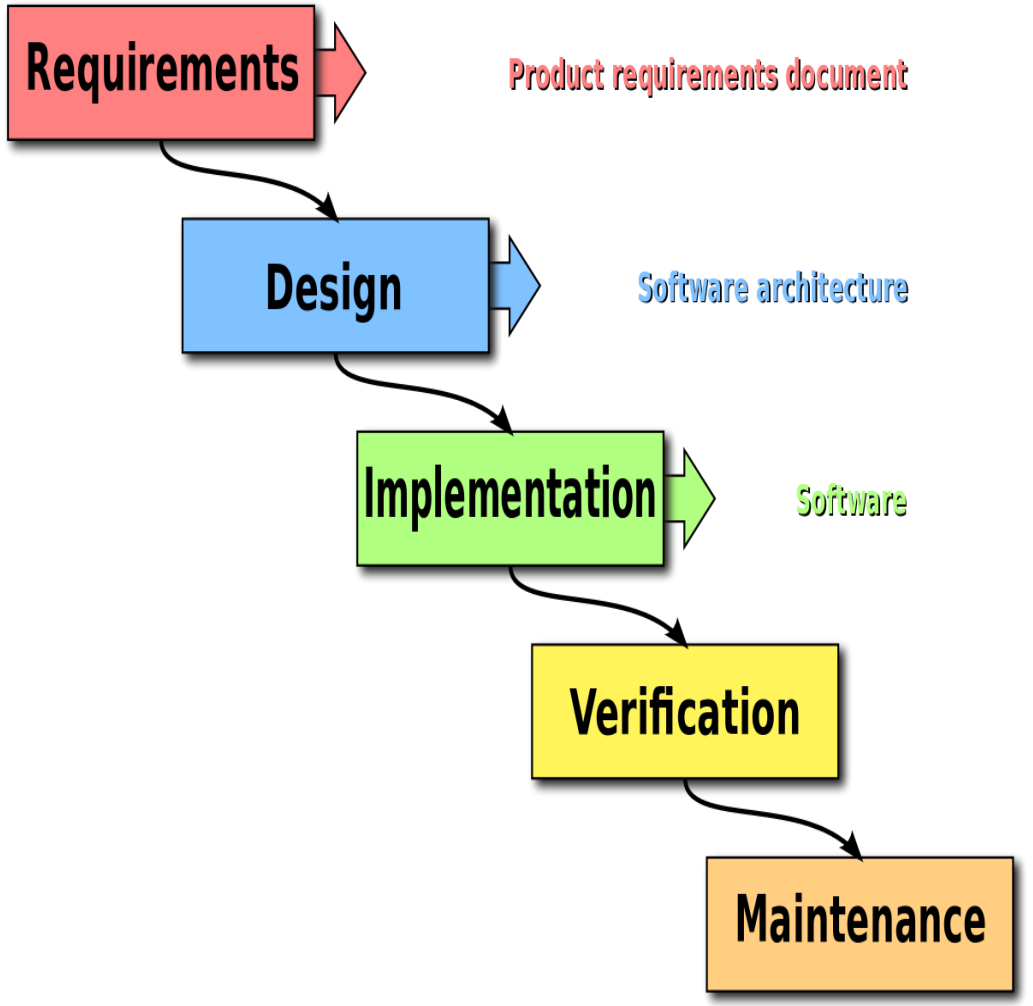
WATERFALL MODEL

- ✓ The first formal description of the waterfall model is often cited as a 1970 article by Winston W.Royce.
- ✓ Royce presented this model as an example of a flawed, non-working model.
- ✓ It has been widely used for software projects ever since.

Royce, Winston (1970), "Managing the Development of Large Software Systems"

WATERFALL MODEL

- ✓ It is also called as linear sequential model.
- ✓ In this model whole application is developed in a sequential approach.
- ✓ In this model each phase must be completed fully before the next phase begin.
- ✓ does not define the process to go back to the previous phase to handle changes in requirement.
- ✓ Provides structure to inexperienced staff.



When to use Waterfall Model

List of things to consider when using the waterfall:

- ✓ The requirements are clear and frozen.
- ✓ Technology is understood and used by the team in different projects.
- ✓ The project cannot be delivered in an iterative manner.
- ✓ Documentation is essential.
- ✓ Professional Project management skills.
- ✓ The project cost is defined.

Waterfall Model - Advantages

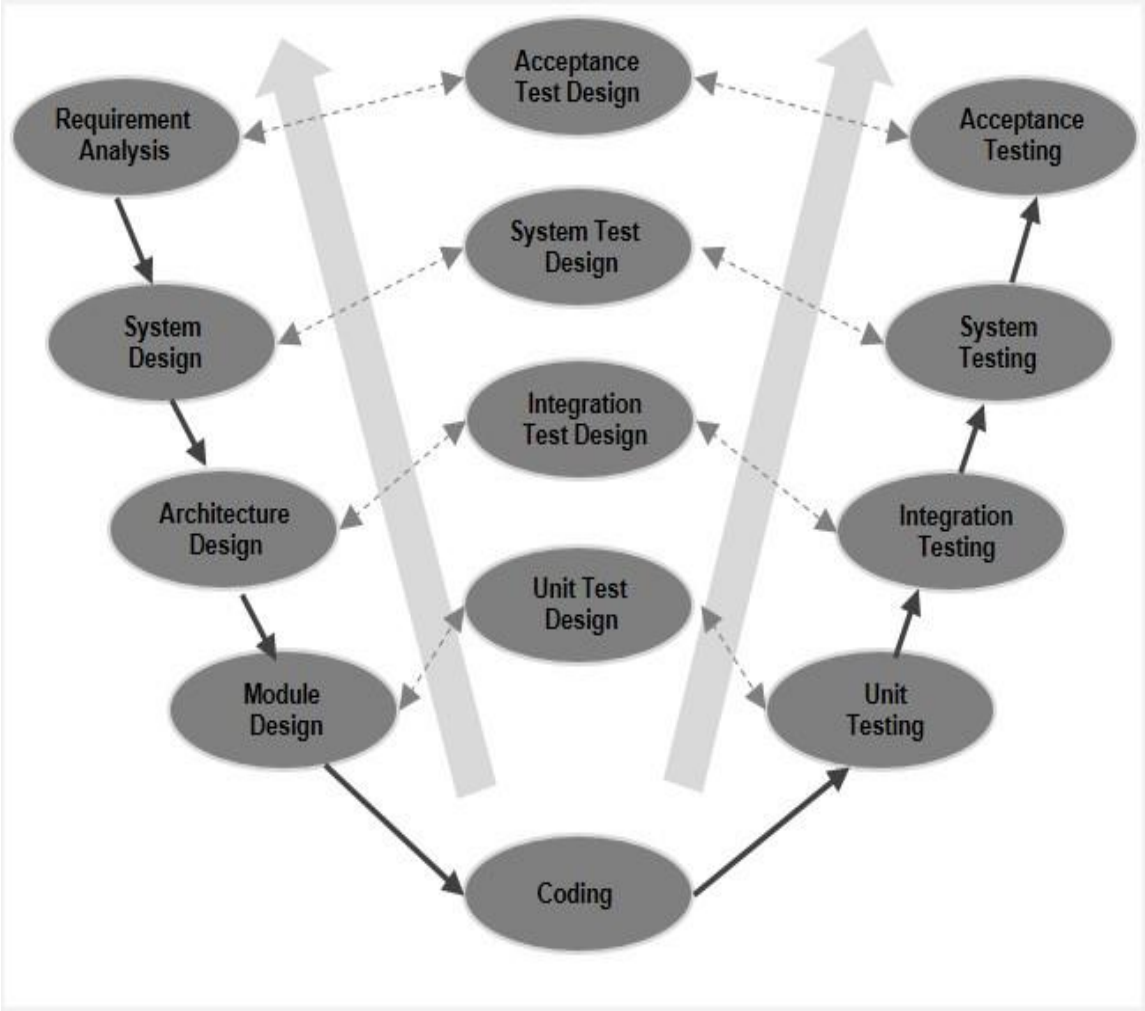
- ✓ It is very easy to explain to the business users and explain the output of each phase.
- ✓ Structured approach.
- ✓ Stages and activities are well defined.
- ✓ It is easier for project managers to plan, schedule the project, utilize the resources, and define the milestones easily.
- ✓ Validation and verification at each phase ensure early detection of errors or misunderstanding at the same phase.
- ✓ Each phase has specific deliverables.

Waterfall Model - Disadvantages

- ✓ It takes the full lifecycle to deliver a workable solution to the customer.
- ✓ It is very difficult to go back to any phase after it finished.
- ✓ It assumes that the requirements of a system can be frozen without any changes or enhancements.
- ✓ A little flexibility and adjusting scope is difficult and expensive.
- ✓ It requires more time for the detailed plan upfront of the project, as the requirements are clear and frozen and it should be visible to have the detailed plan delivered to the customer.
- ✓ It delays the testing phase which can discover a lot of issues in requirements, design, and implementation as well.

V-Shaped Model

- ✓ It is an extension of the waterfall model, Instead of moving down in a linear way, the process steps are bent upwards. after the implementation and coding phase, to form the typical V shape.
- ✓ The major difference between the V-shaped model and waterfall model is the early test planning in the V-shaped model.



When to use V Model

List of things to consider when using the V model:

- ✓ Requirements are well defined, clearly documented and fixed.
- ✓ Product definition is stable.
- ✓ Technology is not dynamic and is well understood by the project team.
- ✓ There are no ambiguous or undefined requirements.
- ✓ The project is short in nature

V Model - Advantages

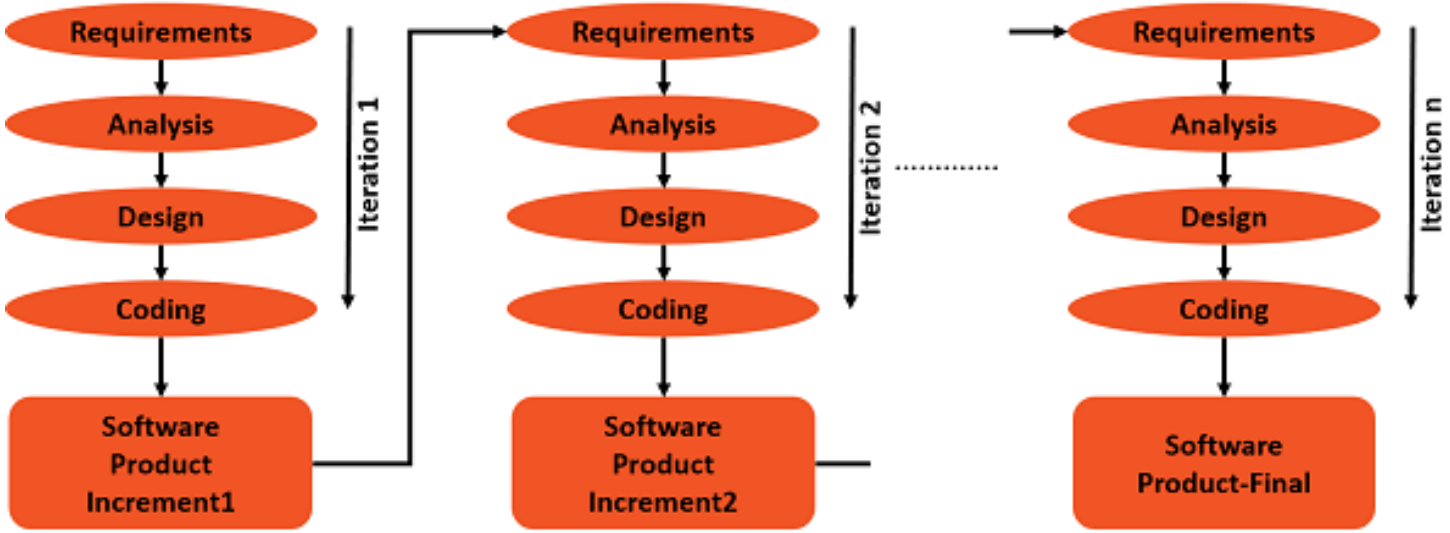
- ✓ Simple and easy to use
- ✓ Each phase has specific deliverables.
- ✓ Higher chance of success over the waterfall model due to the development of test plans early on during the life cycle.
- ✓ Works well for where requirements are easily understood.
- ✓ Verification and validation of the product in the early stages of product development.

V Model - Disadvantages

- ✓ Very inflexible, like the waterfall model.
- ✓ Adjusting scope is difficult and expensive.
- ✓ The software is developed during the implementation phase, so no early prototypes of the software are produced.
- ✓ The model doesn’t provide a clear path for problems found during testing phases.
- ✓ Costly and required more time, in addition to a detailed plan

ITERATIVE INCREMENTAL MODEL

- ✓ developed to overcome the weaknesses of the waterfall model.
- ✓ starts with an initial planning and ends with deployment with the cyclic interactions in between.
- ✓ The basic idea behind this method is to develop a system through repeated cycles (**iterative**) and in smaller portions at a time (**incremental**).
- ✓ allowing software developers to take advantage of what was learned during the development of earlier parts or versions of the system.
- ✓ It can consist of mini waterfalls or mini V-Shaped model



When to use Iterative & Incremental Method

List of things to consider :

- ✓ Most of the requirements are known up-front but are expected to evolve over time.
- ✓ The requirements are prioritized.
- ✓ There is a need to get the basic functionality delivered fast.
- ✓ A project has lengthy development schedules.
- ✓ A project has new technology.
- ✓ The domain is new to the team.

ITERATIVE INCREMENTAL MODEL - ADVANTAGES

- ✓ You can develop prioritized requirements first.
- ✓ Initial product delivery is faster.
- ✓ Customers gets important functionality early.
- ✓ Lowers initial delivery cost.
- ✓ Each release is a product increment, so that the customer will have a working product at hand all the time.
- ✓ Customer can provide feedback to each product increment, thus avoiding surprises at the end of development.
- ✓ Requirements changes can be easily accommodated.

ITERATIVE INCREMENTAL MODEL - DISADVANTAGES

- ✓ Requires effective planning of iterations.
- ✓ Requires efficient design to ensure inclusion of the required functionality and provision for changes later.
- ✓ Requires early definition of a complete and fully functional system to allow the definition of increments.
- ✓ Well-defined module interfaces are required, as some are developed long before others are developed.

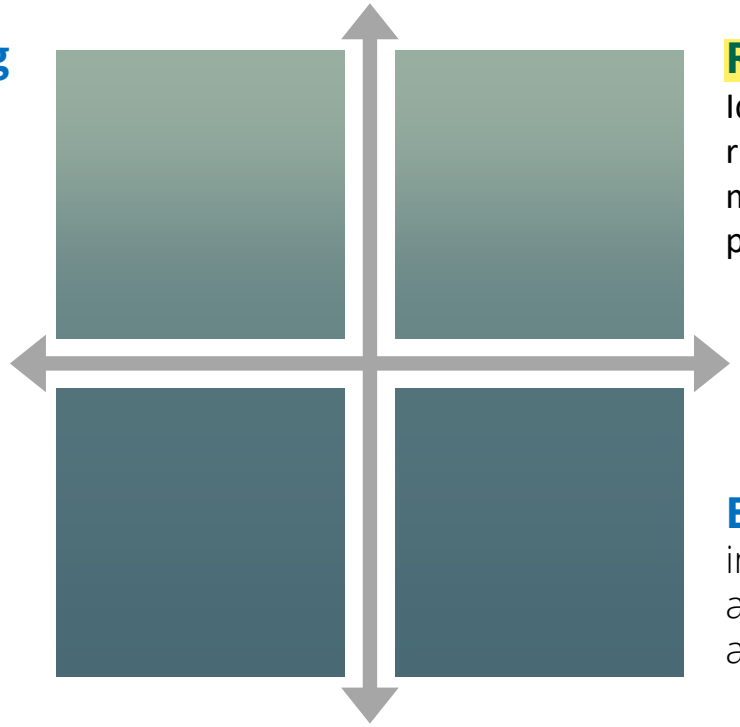
SPIRAL MODEL

- ✓ is a combination of a waterfall model and iterative model.
- ✓ Each phase in spiral model begins with a design goal and ends with the client reviewing the progress.
- ✓ The development team in Spiral-SDLC model starts with a small set of requirement and goes through each development phase for those set of requirements.
- ✓ The software engineering team adds functionality for the additional requirement in every-increasing spirals until the application is ready for the production phase

SPIRAL MODEL

Spiral “areas”

Planning
includes estimating the cost, schedule and resources for the iteration. also involves understanding the system requirements for continuous communication between the system analyst and the customer.

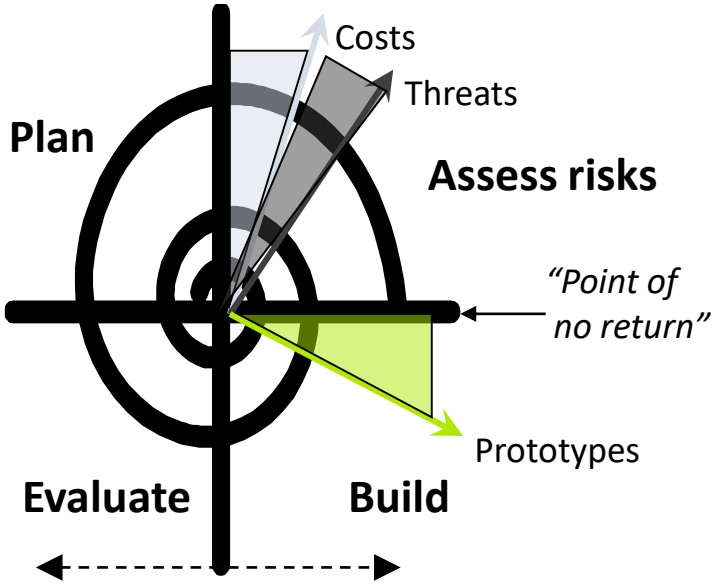


Risk analysis
Identification of potential risk is done while risk mitigation strategy is planned and finalized

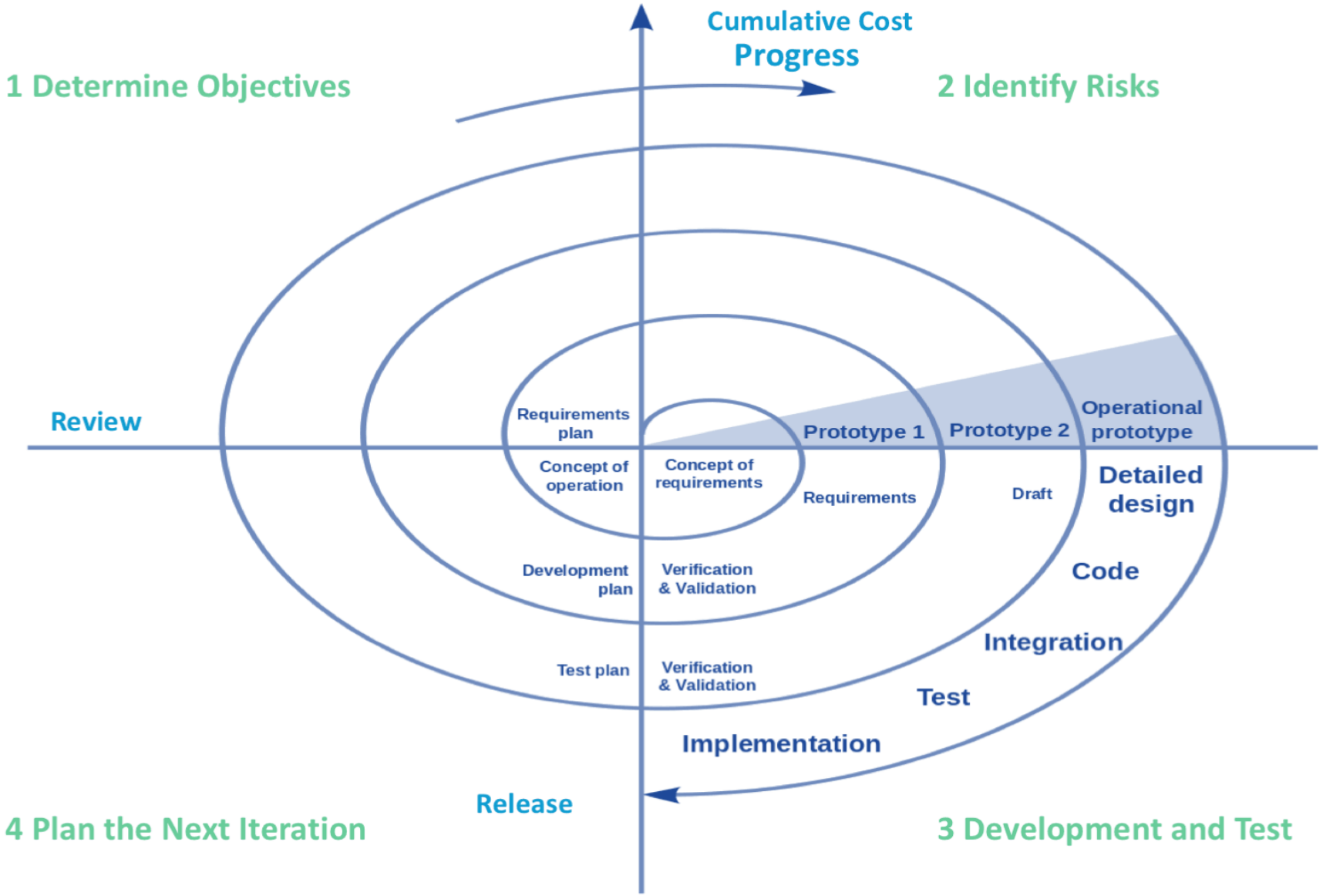
Evaluation
Evaluation of software by the customer. Also, includes identifying and monitoring risks such as schedule slippage and cost overrun

Engineering
includes testing, coding and deploying software at the customer site

Diagrammatically said…



Spiral Model



Spiral Model - Steps

1) Define requirements
   - Through user involvement and analysis of existing system
2) Initial new system design
3) Construct and evaluate an initial prototype
   - Rough (skeletal) system attribute framework
4) Construct a further (refined) prototype
   - Basing it on evaluation of initial prototype
   - Defining its scope
   - Planning its development
   - Implementing it
5) Overall (system-wide) risk assessment
6) Prototype assessment (as per step 4) and possible development of further prototypes
7) Repeat steps 1-5 until refined prototype meets user expectations
8) Construct the system (based on final refined prototype)
9) Test and maintain the system

When to use Spiral Model

List of things to consider when using the Spiral Model:

- ✓ When project is large
- ✓ When releases are required to be frequent
- ✓ When creation of a prototype is applicable
- ✓ When risk and costs evaluation is important
- ✓ For medium to high-risk projects
- ✓ When requirements are unclear and complex
- ✓ When changes may require at any time

Spiral Model - Advantages

- ✓ Additional functionality or changes can be done at a later stage
- ✓ Cost estimation becomes easy as the prototype building is done in small fragments
- ✓ Continuous or repeated development helps in risk management
- ✓ Development is fast and features are added in a systematic way
- ✓ There is always a space for customer feedback

Spiral Model - Disadvantages

- ✓ Risk of not meeting the schedule or budget
- ✓ It works best for large projects only also demands risk assessment expertise
- ✓ For its smooth operation spiral model protocol needs to be followed strictly
- ✓ Documentation is more as it has intermediate phases
- ✓ It is not advisable for smaller project, it might cost them a lot

PROTOTYPING MODEL

"It is easier to tell what you don't like about an existing system than to describe what you would like in an imaginary one"

A.M. Jenkins, 1983

is the process of quickly putting together a **working model** (a prototype) in order to test various aspects of a design, illustrate ideas or features and gather early user feedback.

IEEE defines prototyping as " A type of development in which emphasis is placed on developing prototypes early in the development process to permit early feedback and analysis in support of the development process."

PROTOTYPING MODEL - TYPES

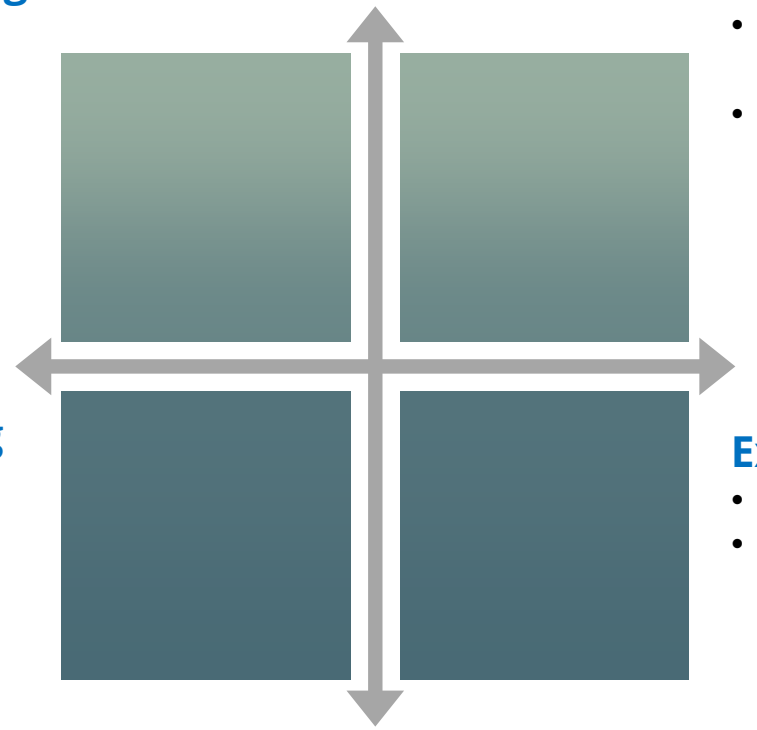
Major software prototyping types

Throwaway/Rapid Prototyping

- uses very little efforts with minimum requirement analysis to build a prototype.
- Once the actual requirements are understood, the prototype is discarded and the actual system is developed with a much clear understanding of user requirements.

Incremental Prototyping

- Incremental prototyping refers to building multiple functional prototypes of the various sub-systems and then integrating all the available prototypes to form a complete system.



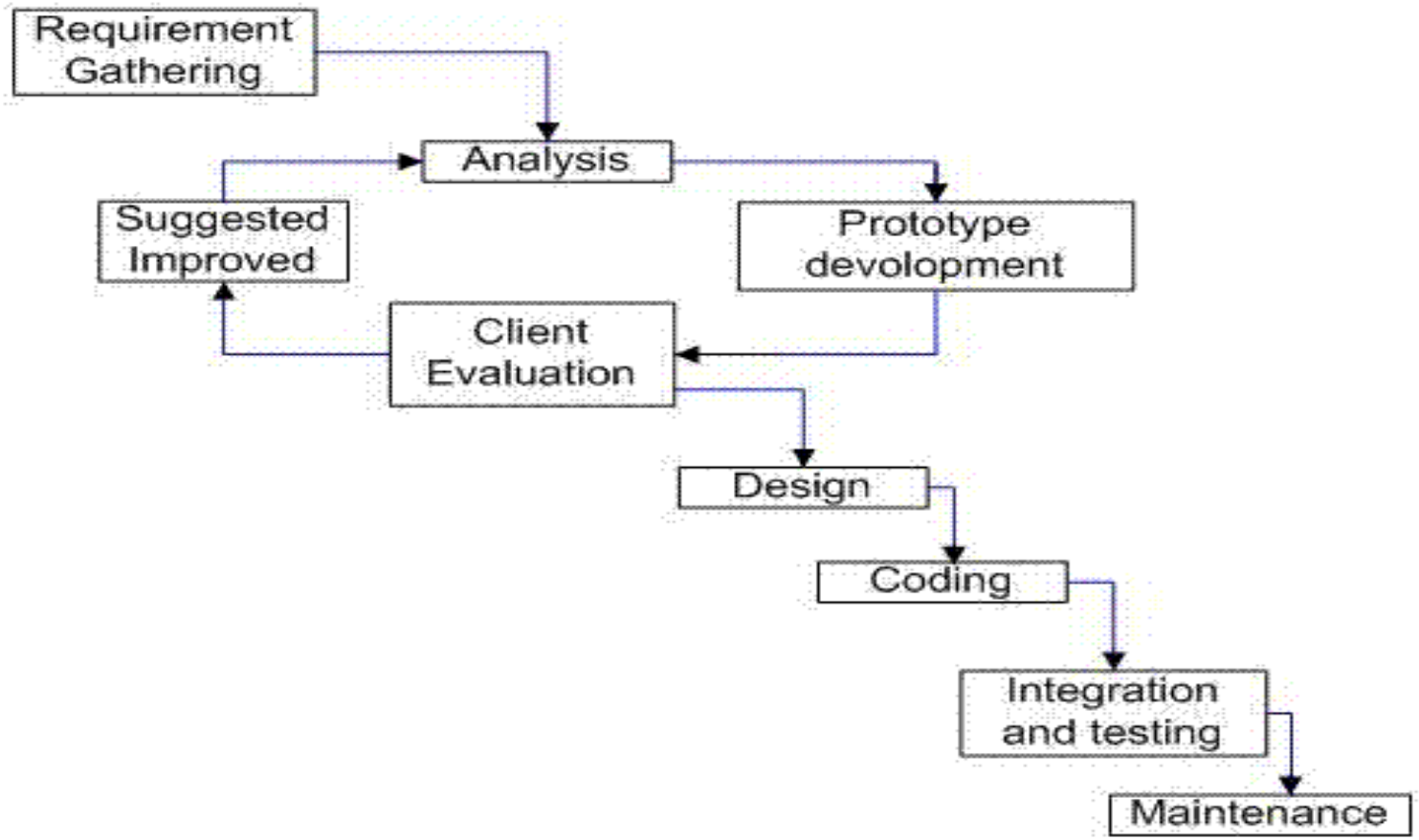
Evolutionary Prototyping

- is based on building actual functional prototypes with minimal functionality in the beginning.
- Only the well-understood requirements are included in the prototype and the requirements are added as and when they are understood.

Extreme Prototyping

- is used in the web development domain.
- consists of three sequential phases.
  - First, a basic prototype with all the existing pages is presented in the HTML format.
  - Then the data processing is simulated using a prototype services layer.
  - Finally, the services are implemented and integrated to the final prototype.

PROTOTYPING MODEL



Evolutionary Prototyping Model

When to use Prototyping Model

List of things to consider when using the Prototyping Model:

- ✓ the requirements are unclear
- ✓ if requirements are changing quickly.
- ✓ can be successfully used for developing user interfaces, high technology software-intensive systems, and systems with complex algorithms and interfaces.
- ✓ a very good choice to demonstrate the technical feasibility of the product.

PROTOTYPING MODEL - ADVANTAGES

- ✓ errors can be detected in the initial stage of the software development process.
- ✓ Missing functionality can be identified, which helps to reduce the risk of failure
- ✓ Customer satisfaction exists because the customer can feel the product at a very early stage.
- ✓ As customers are involved from the early stage, there will be hardly any chance of software rejection.
- ✓ Quicker user feedback helps you to achieve better software development solutions.
- ✓ It is a straightforward model, so it is easy to understand.
- ✓ No need for specialized experts to build the model
- ✓ The prototype serves as a basis for deriving a system specification.
- ✓ The prototype helps to gain a better understanding of the customer's needs.
- ✓ Prototypes may offer early training for future users of the software system.

Prototyping Model - Disadvantages

- ✓ There may be too much variation in requirements each time the prototype is evaluated by the customer.
- ✓ Poor Documentation due to continuously changing customer requirements.
- ✓ It is very difficult for the developers to accommodate all the changes demanded by the customer.
- ✓ There is uncertainty in determining the number of iterations that would be required before the prototype is finally accepted by the customer.
- ✓ After seeing an early prototype, the customers sometimes demand the actual product to be delivered soon.
- ✓ Developers in a hurry to build prototypes may end up with sub-optimal solutions.
- ✓ The customer might lose interest in the product if he/she is not satisfied with the initial prototype.

PROTOTYPING MODEL

Do we need prototyping??

Two “points of interest” for companies to adopt prototyping based methodologies are:

Point 1: They allow us to reduce the cost and time-to-market of a system.

Point 2: For companies building critical systems, prototyping would help them perform formal verification when required. These methodologies provide high level of reliability in the system design and implementation.

What Does it mean?

A-gil-i-ty (ə-'ji-lə-tē) Property consisting of quickness, lightness, and ease of movement; To be very nimble

- The ability to create and respond to change in order to profit in a turbulent business
- The ability to quickly reprioritize use of resources when requirements, technology, and
- Use of evolutionary, incremental, and iterative delivery to converge on an optimal customer
- Maximizing **BUSINESS VALUE** with right sized, just enough, and just-in-time processes and

Highsmith, J. A. (2002). Agile software development ecosystems. Boston, MA: Addison-Wesley.

AGILE DEVELOPMENT

- ✓ Agile is a set of practices, values, and principles for software product development.
- ✓ The core ideas in Agile Development:
  - ➢ Adaptive
  - ➢ Iterative/incremental
  - ➢ People-oriented

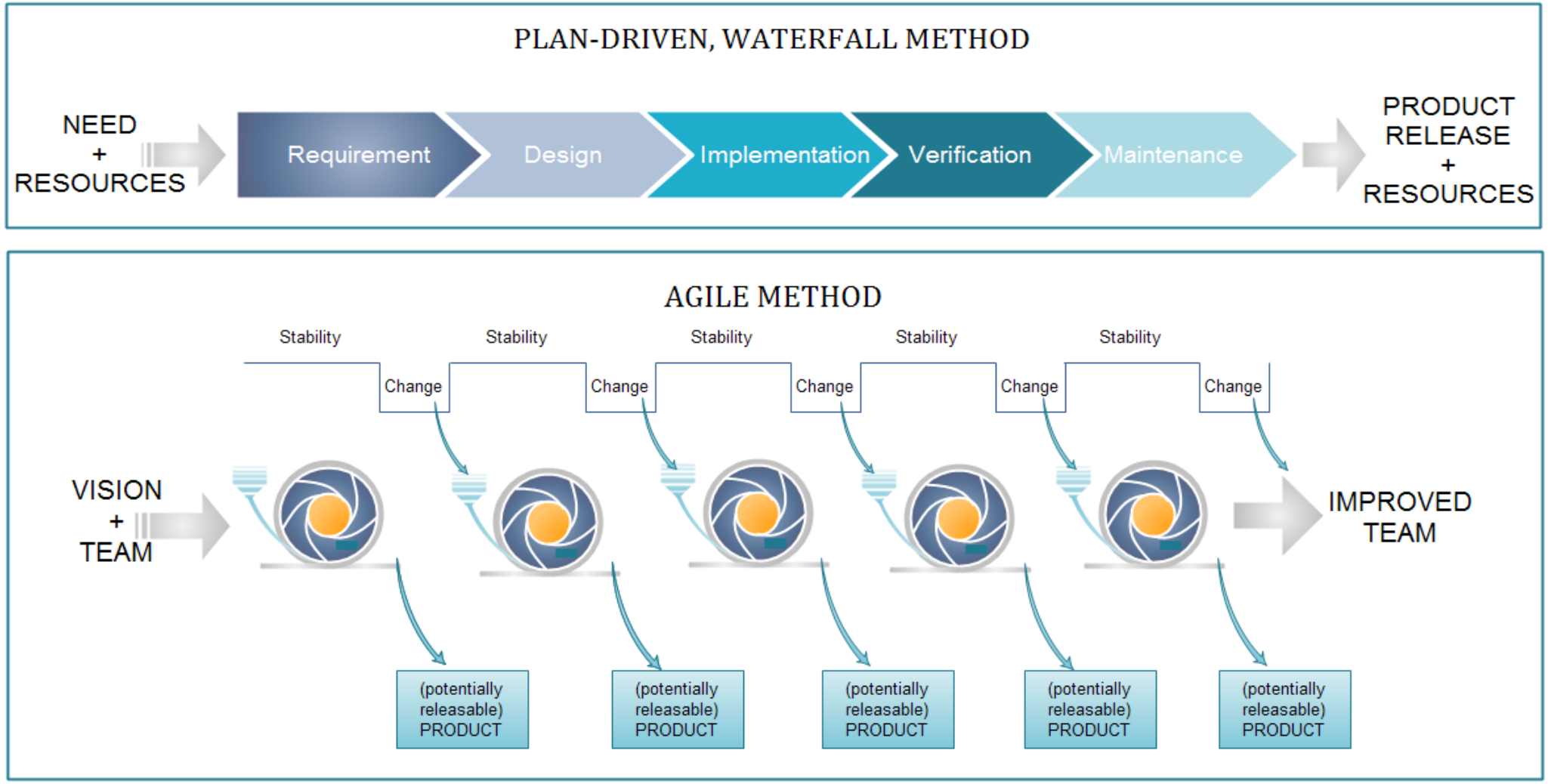
Adaptive means that the teams and the process should be flexible in the presence of “rapid-fire change”.

Iterative and incremental means that Agile Development produces working products in stages – a growing set of “completed and working software”.

People-oriented means the team organization and processes will support good people, who are the most important ingredient to project success.

AGILE DEVELOPMENT

Agile focuses on keeping the process lean and creating minimum viable products (MVPs) that go through a number of iterations before anything is final.



AGILE DEVELOPMENT

Common myth:

The manifesto is often misinterpreted to mean

no documentation, no process, and no plan!

AGILE DEVELOPMENT

Agile Manifesto Principles:

1. Highest priority is satisfy the customer through early and continuous delivery of software.
2. Welcome changing requirements, even late in development…
3. Deliver working software frequently, from a couple of weeks to a couple of months…
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Provide environment and support they need…
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development…a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

When to use Agile

- ✓ When new changes are needed to be implemented and you need to get the project delivered in a short amount of time
- ✓ your project involves iterative, or cyclical, processes in which incremental results will add value for your project .
- ✓ If you do have some budget flexibility
- ✓ trying to build something innovative that does not exist
- ✓ High Product Owner Involvement.
- ✓ your organization doesn't have strict processes to follow and you have the luxury of being able to work flexibly

AGILE DEVELOPMENT - ADVANTAGE

- ✓ Customer satisfaction by rapid, continuous delivery of useful software.
- ✓ People and interactions are emphasized rather than process and tools.
- ✓ Customers, developers and testers constantly interact with each other.
- ✓ Working software is delivered frequently (weeks rather than months).
- ✓ Face-to-face conversation is the best form of communication.
- ✓ Close, daily cooperation between business people and developers.
- ✓ Continuous attention to technical excellence and good design.
- ✓ Regular adaptation to changing circumstances.
- ✓ Even late changes in requirements are welcomed

AGILE DEVELOPMENT - DISADVANTAGE

- ✓ In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- ✓ There is lack of emphasis on necessary designing and documentation.
- ✓ 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.

AGILE DEVELOPMENT - SCRUM

- ✓ Scrum is an agile framework that allows us to focus on delivering the highest business value in the shortest time.
- ✓ It allows us to rapidly and repeatedly inspect actual working software (every two weeks to one month).
- ✓ The business sets the priorities. Teams self-organize to determine the best way to deliver the highest priority features.
- ✓ Every two weeks to a month anyone can see real working software and decide to release it as is or continue to enhance it for another sprint.

AGILE DEVELOPMENT - XP

- ✓ **Extreme Programming (XP),** originally described by Kent Beck
- ✓ is a disciplined approach to delivering high-quality software quickly and continuously.
- ✓ is intended to improve software quality and responsiveness in the face of changing customer requirements
- ✓ original XP method is based on four simple values – simplicity, communication, feedback, courage.
- ✓ Also has twelve supporting practices:

1. Planning Game
2. Small Releases
3. Customer Acceptance Tests
4. Simple Design
5. Pair Programming
6. Test-Driven Development
7. Refactoring
8. Continuous Integration
9. Collective Code Ownership
10. Coding Standards
11. Metaphor
12. Sustainable Pace

AGILE DEVELOPMENT – XP PRACTICES

1. **Planning Game**
   - ✓ Planning for the upcoming iteration and Uses stories provided by the customer
   - ✓ Technical persons determine schedules, estimates, costs, etc
   - ✓ A result of collaboration between the customer and the developers
2. **Small Releases**
   - ✓ Small in terms of functionality
   - ✓ Less functionality means releases happen more frequently
   - ✓ Support the planning game
3. **Metaphor**
   1. The oral architecture of the system
   2. Encourages a common set of terms for the system
   3. A quick and easy way to explain the system
4. **Simple Design**
   1. Do as little as needed, nothing more
   2. Easier to understand what is going on
   3. Helps keeps programmers on track
5. **Testing**
   - ✓ Unit testing - promote testing completeness
   - ✓ Test-first design - gives developers a goal
   - ✓ All automated - gives a suite of regression

AGILE DEVELOPMENT – XP PRACTICES

6. **Refactoring**
   - ✓ Changing how the system does something but not what is done
   - ✓ Improves the quality of the system in some way and Increases developer knowledge of the system
7. **Pair Programming**
   - ✓ Two Developers, One monitor, One Keyboard
   - ✓ One “drives” and the other thinks
   - ✓ Switch roles as needed
   - ✓ Two people are more likely to answer the following questions:
     I. Is this whole approach going to work?
     II. What are some test cases that may not work yet?
     III. Is there a way to simplify this?
8. **Collective Ownership**
   - ✓ The idea that all developers own all of the code
   - ✓ Enables refactoring
   - ✓ Helps mitigate the loss of a team member leaving
   - ✓ Promotes developers to take responsibility for the system as a whole rather then parts of the system
9. **Continuous Integration**
   - ✓ New features and changes are worked into the system immediately
   - ✓ Code is not worked on without being integrated for more than a day

AGILE DEVELOPMENT – XP PRACTICES

10. 40-Hour Workweek

- ✓ The work week should be limited to 40 hours
- ✓ Regular overtime is a symptom of a problem and not a long term solution
- ✓ Most developers lose effectiveness past 40-Hours
- ✓ Value is placed on the developers well-being
- ✓ Management is forced to find real solutions

11. On-site Customer

- ✓ Gives quick and continuous feedback to the development team
- ✓ Can give quick and knowledgeable answers to real development questions
- ✓ Makes sure that what is developed is what is needed
- ✓ Functionality is prioritized correctly

12. Coding Standards

- ✓ All code should look the same
- ✓ It should not possible to determine who coded what based on the code itself
- ✓ Reduces the amount of time developers spend reformatting other peoples' code

AGILE DEVELOPMENT - LEAN

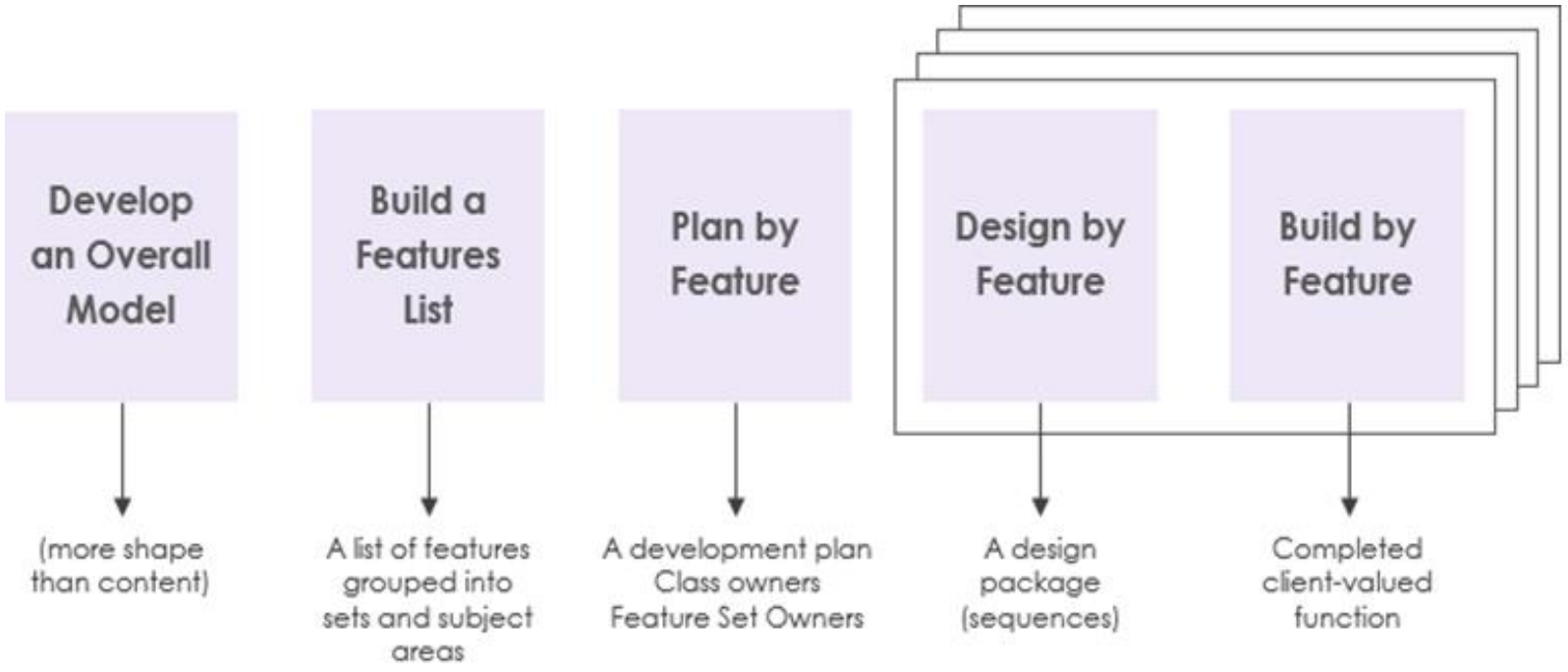
- ✓ Lean originated with the Toyota Production System, which revolutionized the manufacture of physical goods in the 1950s, ‘60s, and beyond
- ✓ Lean maintains its hold in manufacturing but has also found new applications in knowledge work, helping businesses in all industries *eliminate waste, improve processes, and boost innovation*.
- ✓ Software development is a natural application of Lean methodology because, much like manufacturing, it generally follows a defined process, has some defined conditions of acceptance, and results in the delivery of tangible value.
- ✓ key concepts that guide all practice of Lean methodology, which we call the Pillars of Lean. They are:
  - ➢ Continuous improvement
  - ➢ Respect for people
  - ➢ Lightweight Leadership

AGILE DEVELOPMENT - KANBAN

- ✓ Kanban is a highly visual workflow management method that is popular among Lean teams.
- ✓ In fact, 83% of teams practicing Lean use Kanban to visualize and actively manage the creation of products with an emphasis on continual delivery.
- ✓ Kanban is a process designed to help teams work together more effectively.
- ✓ Kanban is based on 3 basic principles:
  - ✓ Visualize what you'll do today (workflow): Seeing all the items within the context of each other can be very informative
  - ✓ Limit the amount of work in progress (WIP): This helps balance the flow-based approach so teams don't start and commit to too much work at once
  - ✓ Enhance flow: When something is finished, the next highest priority item from the backlog is pulled into play

AGILE DEVELOPMENT - FDD

- ✓ is an iterative and incremental software development process and is an agile method for developing software.
- ✓ allows teams to update the project regularly and identify errors quickly.
- ✓ practices are driven from a client-valued functionality (feature) perspective
- ✓ main purpose is to deliver tangible, working software repeatedly in a timely manner.
- ✓ It consists of five basic activities:
  - ➢ Development of an overall model
  - ➢ Building of a feature list
  - ➢ Planning by feature
  - ➢ Designing by feature
  - ➢ Building by feature.



AGILE DEVELOPMENT - DSDM

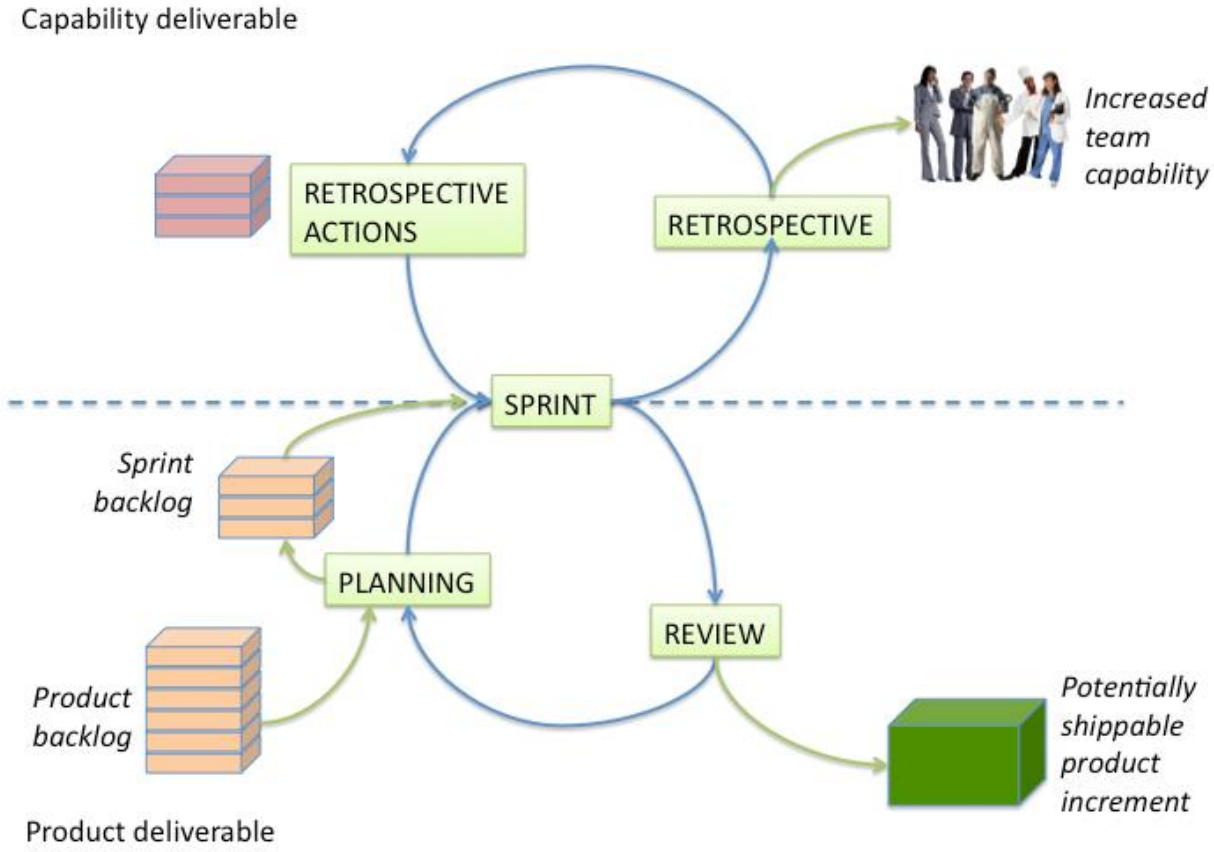
- ✓ Dynamic Systems Development Method (DSDM)
- ✓ is a framework that is made up of eight principles, several best practice techniques.
- ✓ is a framework that prioritizes schedule and quality over functionality, which fixes cost, quality and time at the start and uses the **MoSCoW** method of prioritization, which breaks a project down into four different types of requirements:
    1. Must have (M)
    2. Should have (S)
    3. Could have (C)
    4. Won't have (W)
- ✓ Eight principles of DSDM 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

AGILE DEVELOPMENT - CRYSTAL

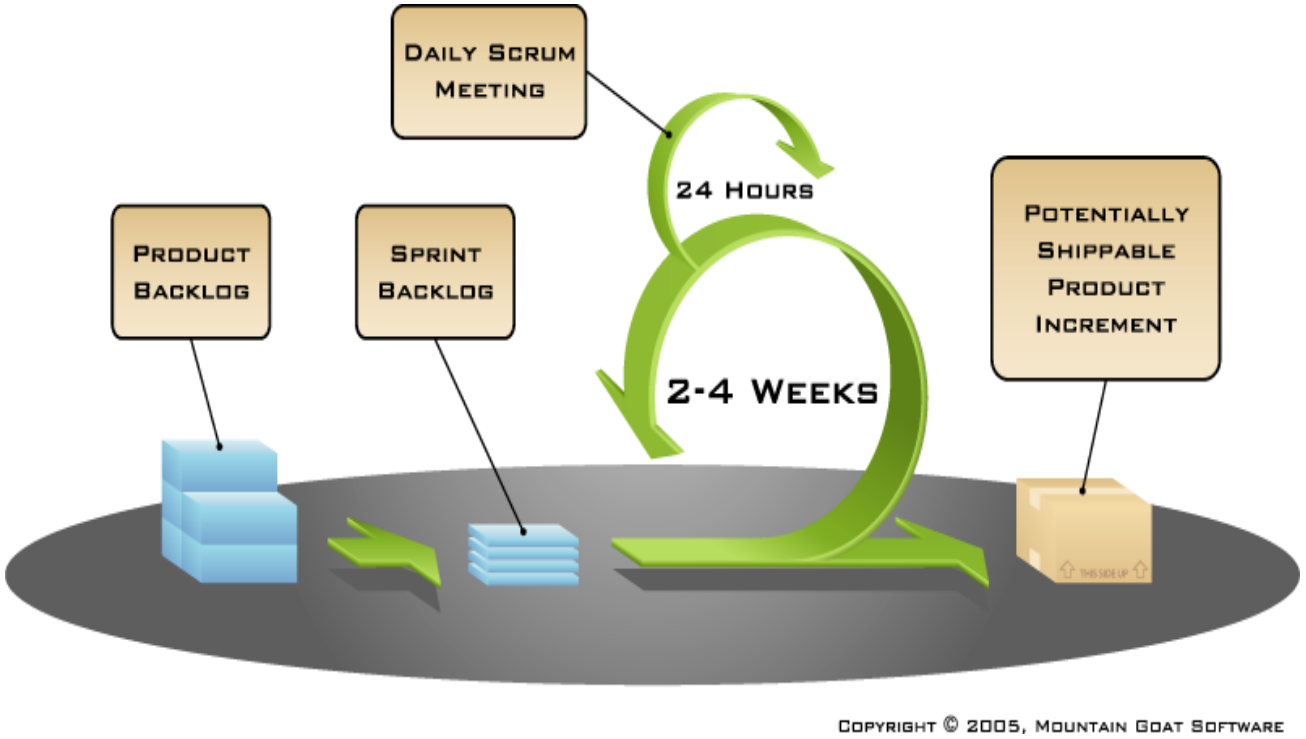
- ✓ Crystal methods are a family of methodologies (the Crystal family) that were developed by Alistair Cockburn in the mid-1990s.
- ✓ Crystal methods are focused on:
    1. People
    2. Interaction
    3. Community
    4. Skills
    5. Talents
    6. Communications

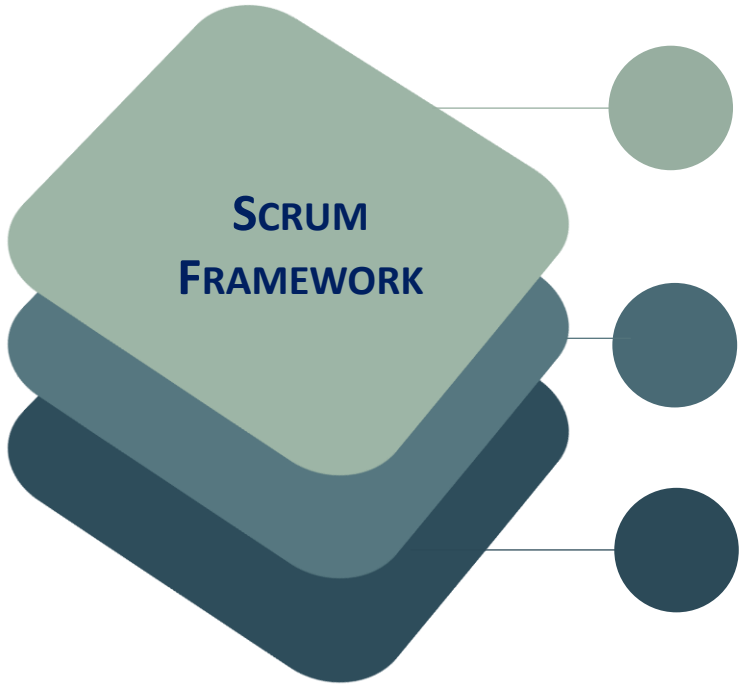
SCRUM

- ✓ A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.
- ✓ Scrum is:
  - ➢ Lightweight
  - ➢ Simple to understand
  - ➢ Difficult to master



More on Scrum





Events

1. Sprint planning
2. Sprint review
3. Sprint retrospective
4. Daily scrum meeting

Roles

1. Product owner
2. ScrumMaster
3. Team

Artifacts

1. Product backlog
2. Sprint backlog
3. Definition of Done
4. Burndown charts

SCRUM - ROLES

✓ Product owner

- Clearly expressing Product Backlog items
- Ordering the items in the Product Backlog to best achieve goals and missions
- Optimizing the value of the work the Development Team performs
- Ensuring that the Product Backlog is visible, transparent, and clear to all, and shows what the Scrum Team will work on next
- Ensuring the Development Team understands items in the Product Backlog to the level needed.

✓ Team

- The Development Team consists of professionals who do the work of delivering a potentially releasable Increment of "Done" product at the end of each Sprint.
- Development Teams have the following characteristics:
  - They are self-organizing. No one (not even the Scrum Master) tells the Development Team
  - how to turn Product Backlog into Increments of potentially releasable functionality;
  - Scrum recognizes no titles for Development Team members, regardless of the work being performed by the person;
  - Scrum recognizes no sub-teams in the Development Team, regardless of domains that need to be addressed like testing, architecture, operations, or business analysis; and,
  - Individual Development Team members may have specialized skills and areas of focus, but accountability belongs to the Development Team as a whole.

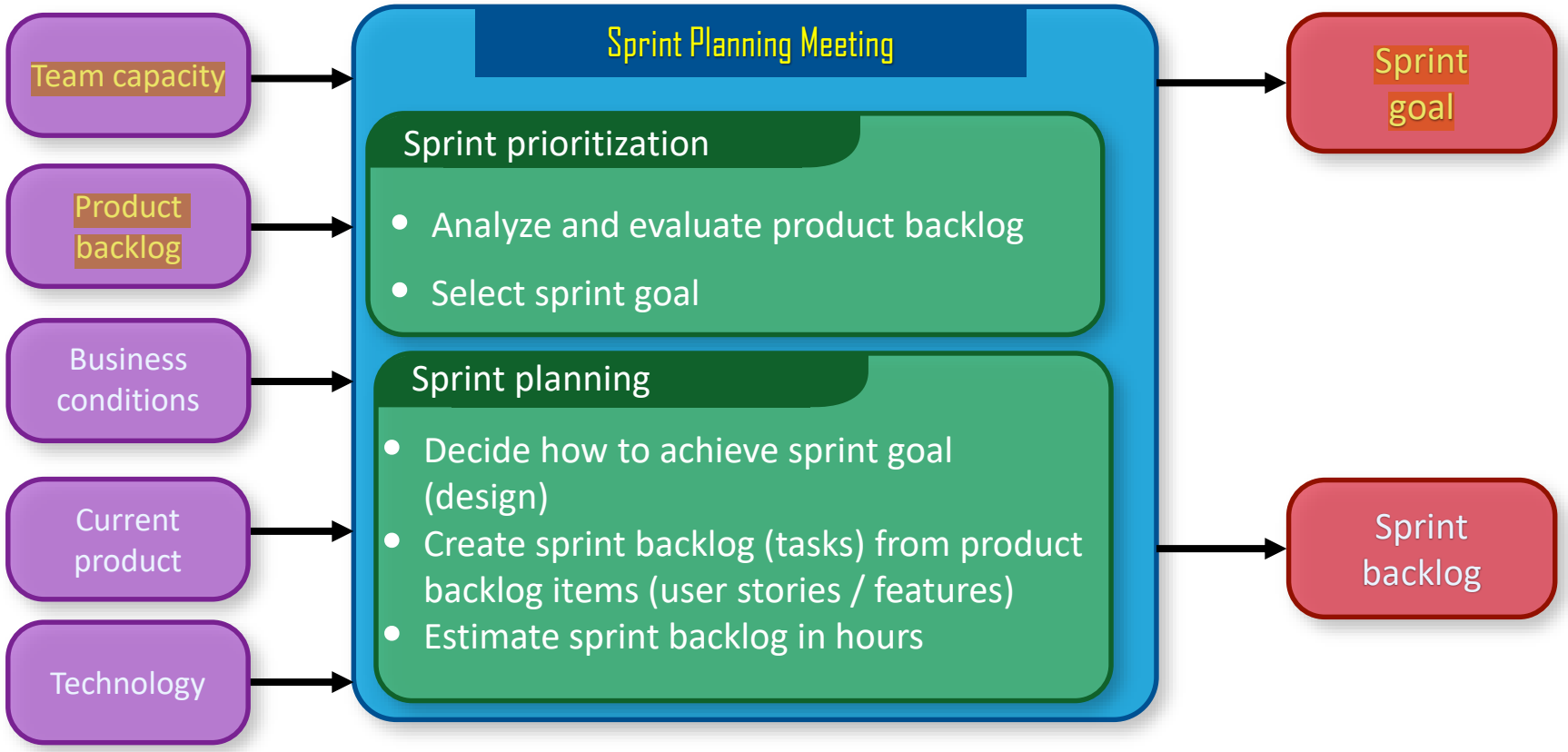
SCRUM - ROLES

- ✓ **ScrumMaster**
  - ➢ **Scrum Master Service to the Product Owner**
    - Ensuring that goals, scope, and product domain are understood by everyone on the Scrum
    - Finding techniques for effective Product Backlog management
    - Helping the Scrum Team understand the need for clear and concise Product Backlog items;
    - Understanding and practicing agility; and,
    - Facilitating Scrum events as requested or needed.
  - ➢ **Scrum Master Service to the Development Team**
    - Coaching the Development Team in self-organization and cross-functionality;
    - Helping the Development Team to create high-value products;
    - Removing impediments to the Development Team's progress;
    - Facilitating Scrum events as requested or needed; and,
    - Coaching the Development Team in organizational environments in which Scrum is not yet fully adopted and understood.
  - ➢ **Scrum Master Service to the Organization**
    - Leading and coaching the organization in its Scrum adoption;
    - Planning Scrum implementations within the organization;
    - Helping employees and stakeholders understand and enact Scrum and empirical product development;

SCRUM - EVENTS

✓ Sprint planning

- Sprint Planning answers the following:
  - What can be delivered in the Increment resulting from the upcoming Sprint?
  - How will the work needed to deliver the Increment be achieved?



SCRUM - EVENTS

2 - 4 hours

✓ Sprint review

- Attendees include the Scrum Team and key stakeholders invited by the Product Owner.
- The Product Owner explains what Product Backlog items have been “Done” and what has not been “Done”.
- The Development Team discusses what went well during the Sprint, what problems it ran into, and how those problems were solved.
- The entire group collaborates on what to do next, so that the Sprint Review provides valuable input to subsequent Sprint Planning.
- Review of how the marketplace or potential use of the product might have changed what is the most valuable thing to do next.
- Review of the timeline, budget, potential capabilities, and marketplace for the next anticipated releases of functionality or capability of the product.

SCRUM - EVENTS

4 hours

✓ Sprint retrospective

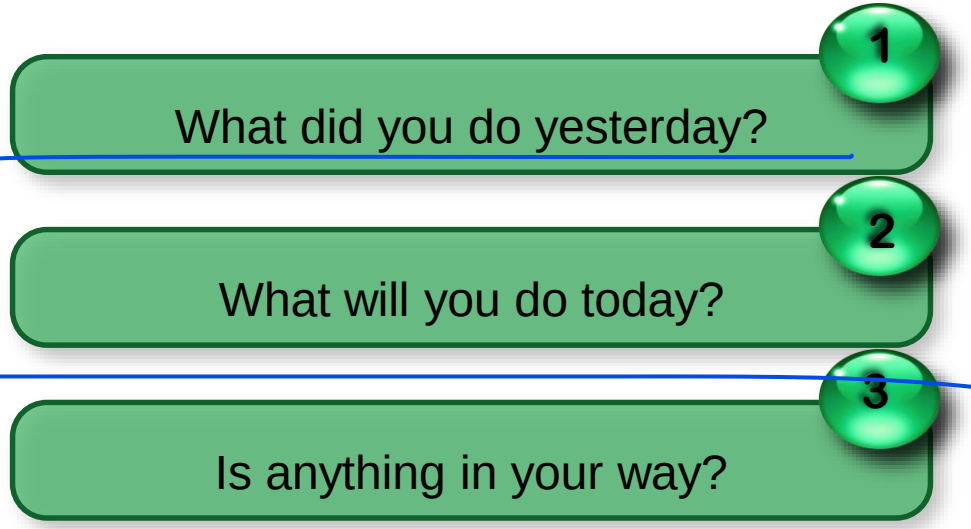
- The purpose of the Sprint Retrospective is to:
  - Inspect how the last Sprint went with regards to people, relationships, process, and tools;
  - Identify and order the major items that went well and potential improvements; and,
  - Create a plan for implementing improvements to the way the Scrum Team does its work..

Whole team gathers and discusses what they'd like to:

BLAME

Scrum - Events

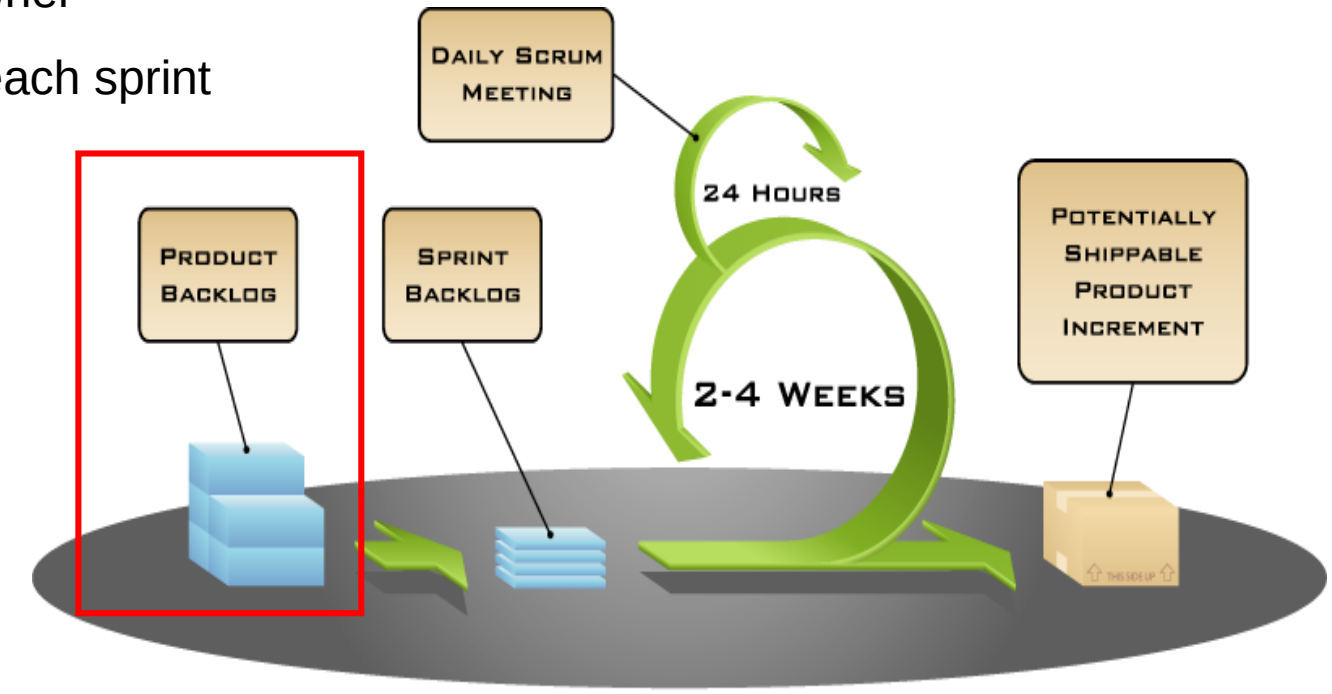
- ✓ **Daily Scrum**
  - Parameters
    - Daily
    - 15-minutes
    - Stand-up
  - Not for problem solving
  - Whole world is invited
  - Only team members, ScrumMaster, product owner, can talk
  - Helps avoid other unnecessary meetings and everyone answer three Questions:



Scrum - Artifacts

✓ Product backlog

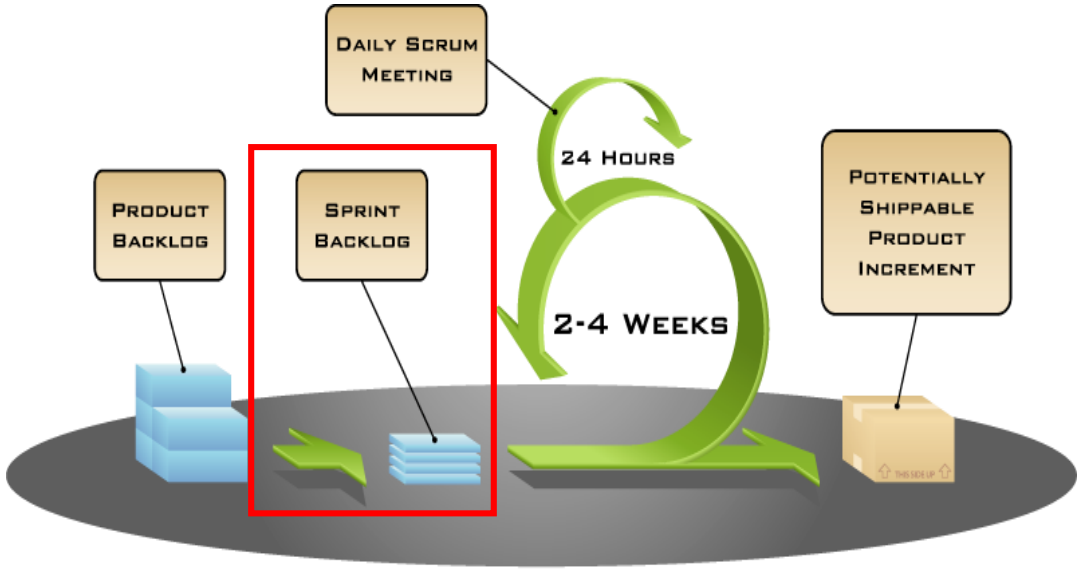
- The requirements
- A list of all desired work on the project
- Ideally expressed such that each item has value to the users or customers of the product
- Prioritized by the product owner
- Reprioritized at the start of each sprint



SCRUM - ARTIFACTS

✓ Sprint backlog

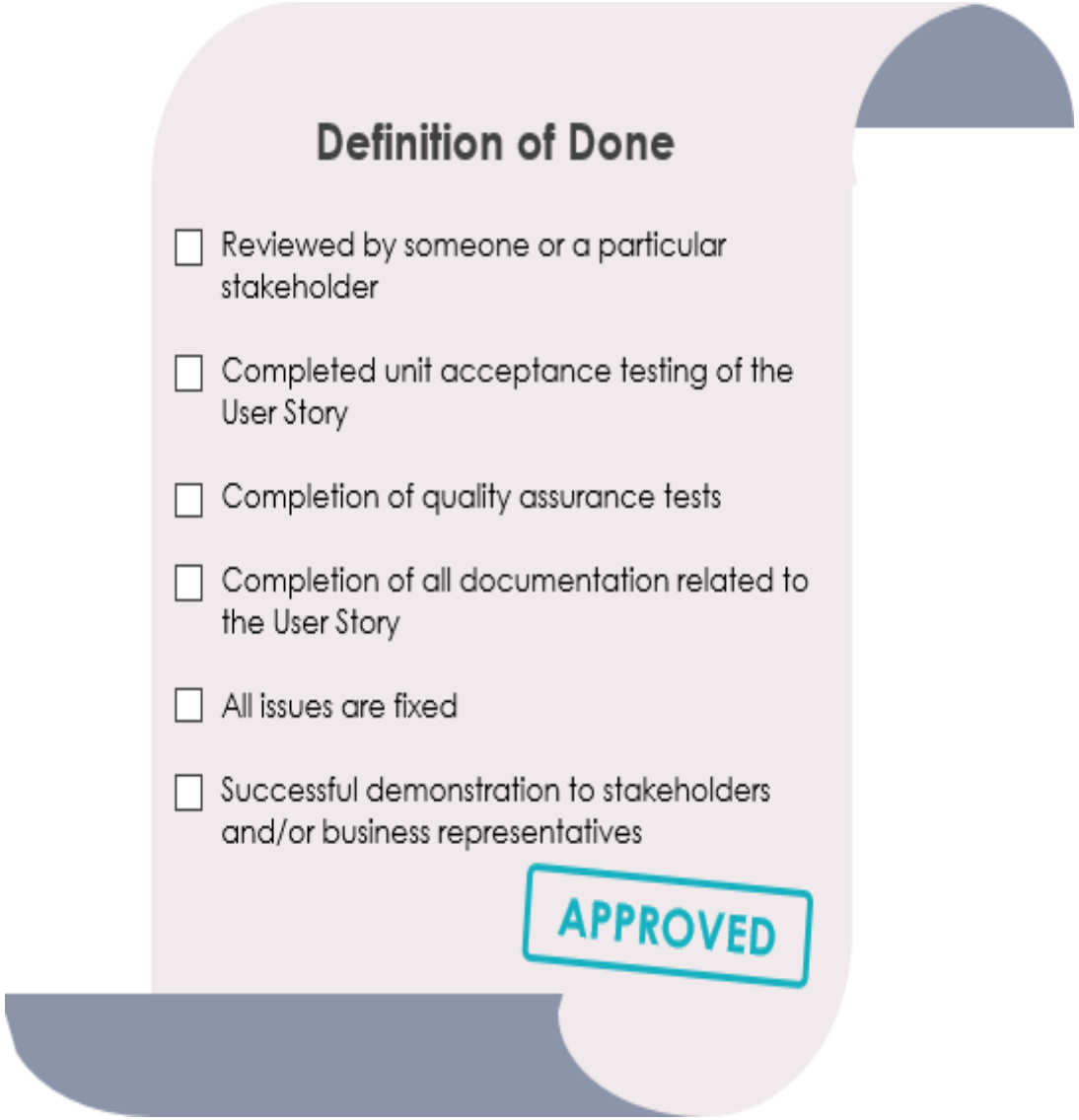
- is the set of Product Backlog items selected for the Sprint
- is a forecast by the Development Team about what functionality will be in the next Increment
- Sprint Backlog will be changed for the following reasons:
- Over time, the development team has a better understanding of the requirements and may find that new tasks need to be added to the Sprint Backlog.
- Bugs of features are added as new tasks, which are all unfinished tasks committed previously of the Sprint.



SCRUM - ARTIFACTS

✓ Definition of Done

- Every Product Backlog item has **acceptance criteria** that define measurably what must be met when the item is declared to be **done.**
- Definition of Done is a shared understanding of the Scrum Team on the meaning of work to be complete.
- It typically contains **quality criteria**, **constraints** and overall **non-functional requirements**.

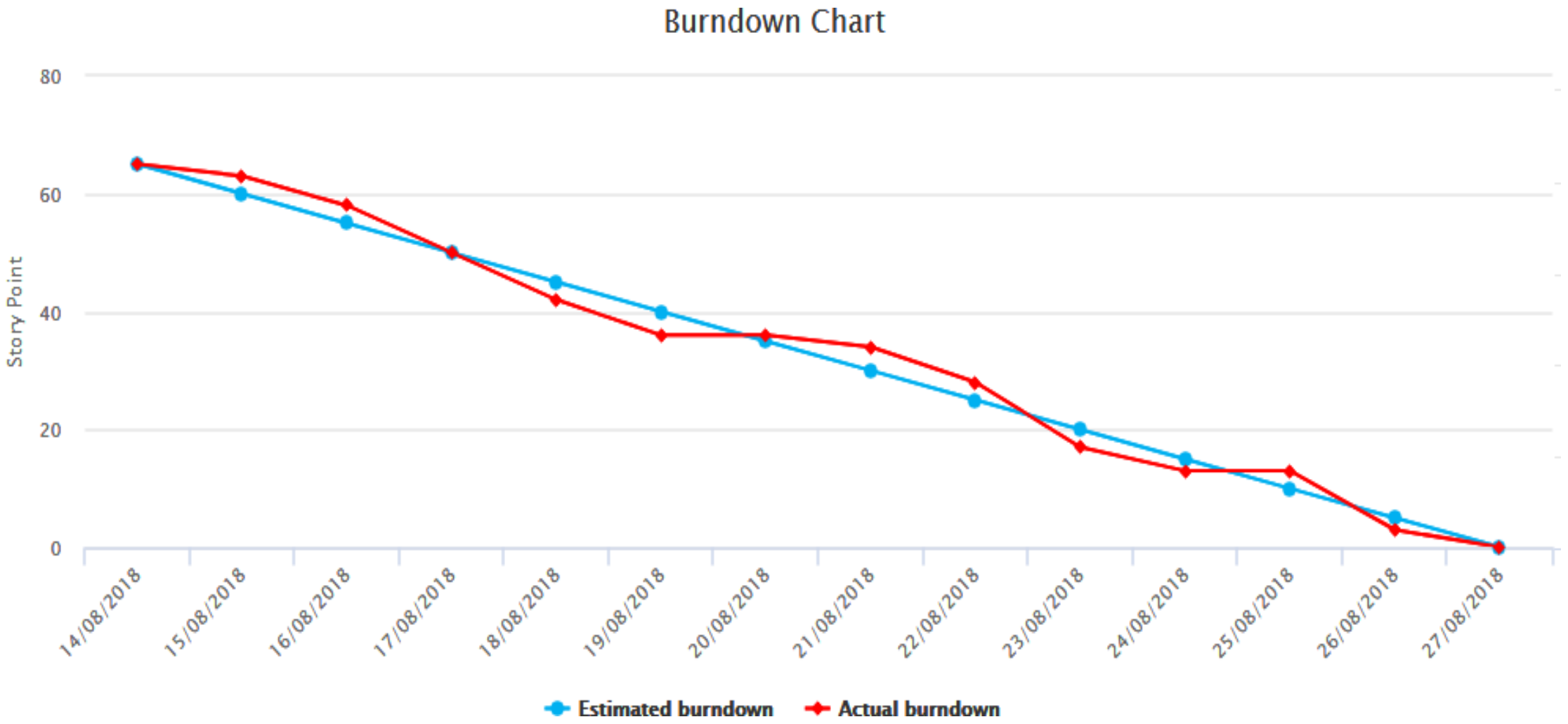


SCRUM - ARTIFACTS

✓ Burndown chart

- are graphs that give an overview of progress over time while completing a project.
- this chart starts with the total number of points the team has taken on for the sprint, and tracks on a day-to-day basis how many of those points have been completed and are ready for the sprint demo.
- As tasks are completed, the graph “burns down” to zero.
- is usually maintained by the scrum master, and may be updated on a daily basis, perhaps after the daily scrum.
- A typical burndown chart starts with a straight diagonal line from the top left to the bottom right, showing an “ideal” burndown rate for the sprint.

SCRUM - ARTIFACTS



STORY POINTS ????

Story points are the effort to do something based on the volume, risk, uncertainty and complexity of the work.

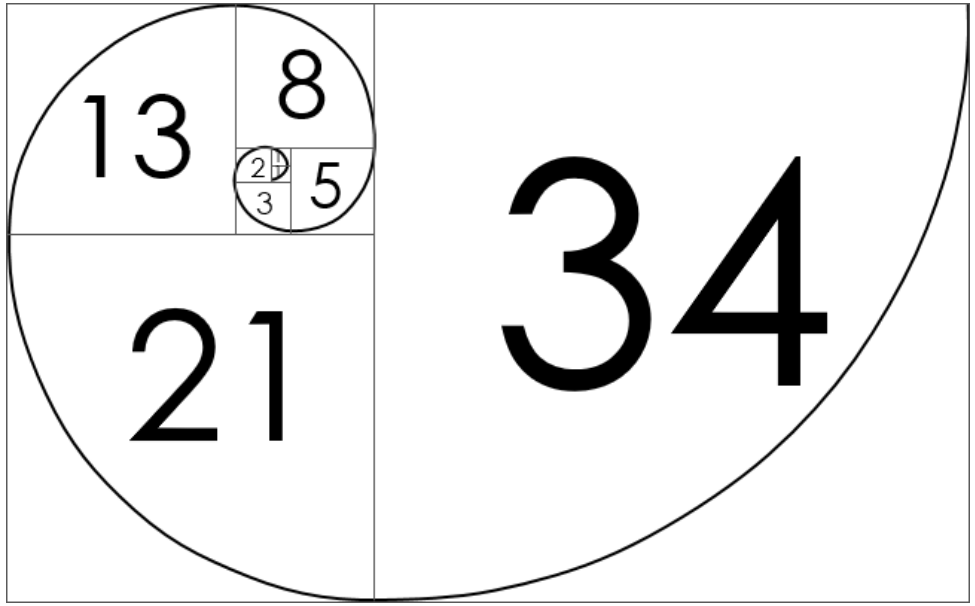
In other words There are some elements considered before assigning the story points to a situation, which are mentioned below:

- ✓ The complexity level of a particular story
- ✓ The potential amount of effort that is required to implement
- ✓ The total number of the unknown affecting and non-affecting factors

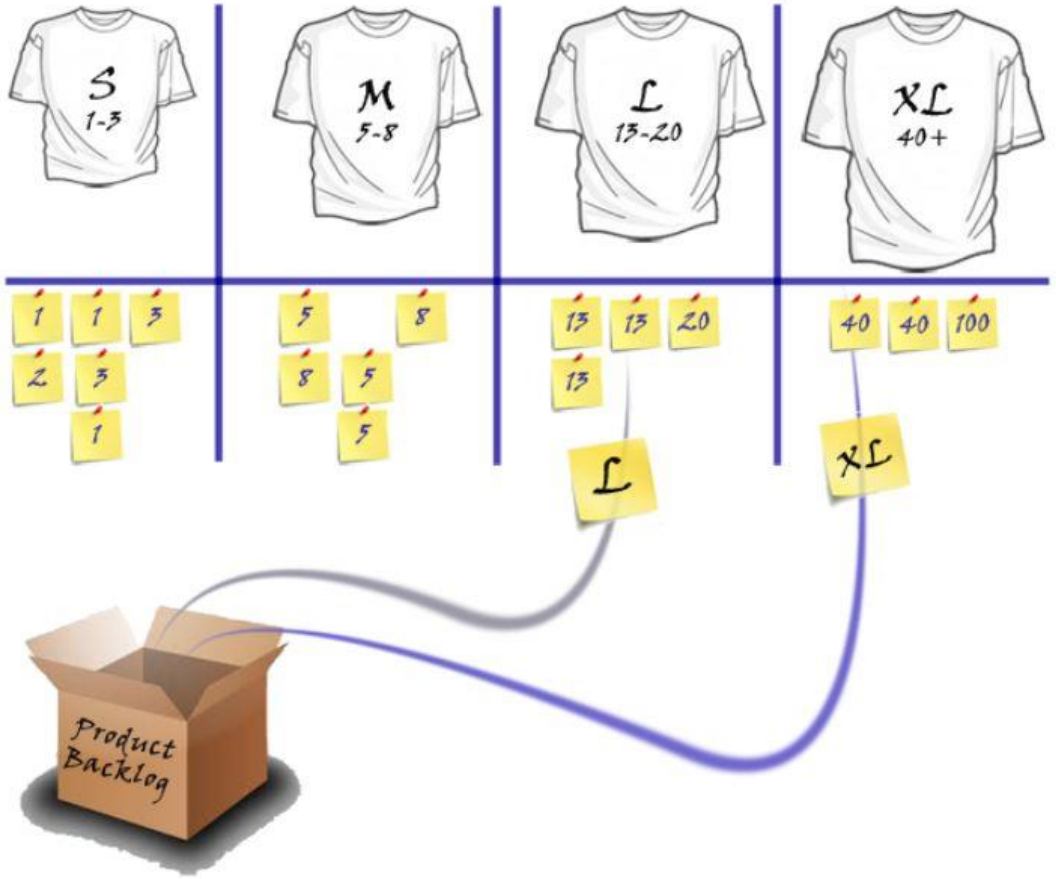
How are Story Points Expressed

The story points can be expressed in two different ways

As Numerical Range in terms of Fibonacci Sequence



As T shirt Size Range in terms of X-S, i.e. Extra Large



QUIZ

Following is the performance of a Scrum team during the first 6 sprints of a project.

1.Sprint: 15 SP
2.Sprint: 5 SP
3.Sprint: 20 SP
4.Sprint: 15 SP
5.Sprint: 25 SP
6.Sprint. 10 SP

Question 1: What is the velocity of the team ?

Question 2: How many Story Points are likely to be achieved by this team in Sprint 7 ?

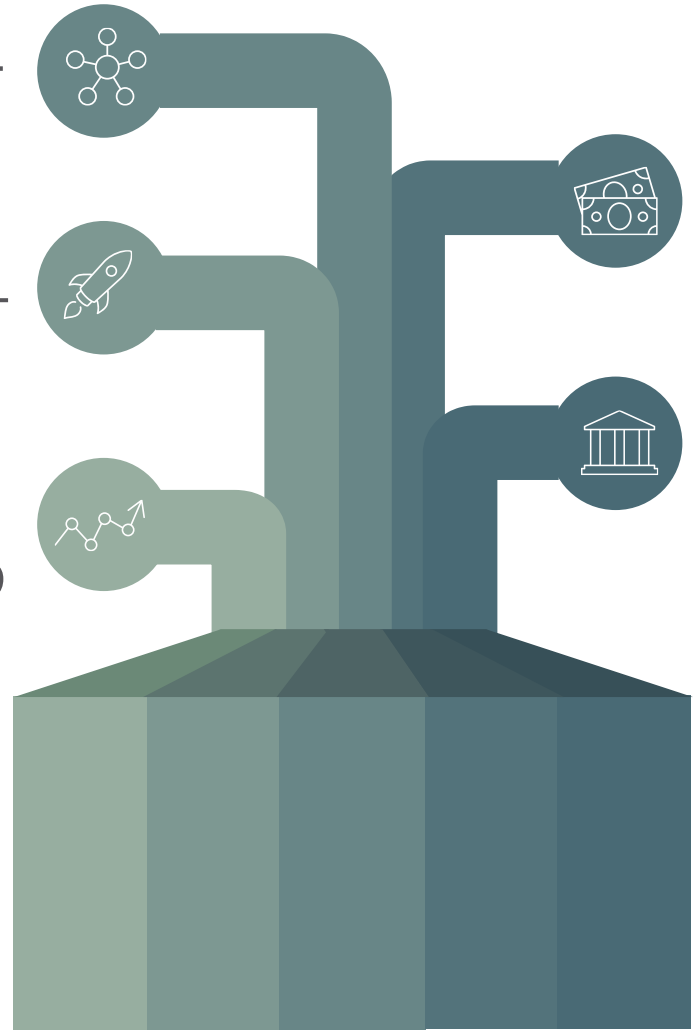
SELECTING RIGHT SDLC

Learn SDLC
01

- Should have proper knowledge of SDLC
- Need to understand those.

Define the Criteria
03

- Is the SDLC suitable for the size of team and their skills?
- Is the SDLC suitable for the selected technology we use for implementing the solution?
- Is the SDLC suitable for client and stakeholders concerns and priorities?
- Is the SDLC suitable for the geographical situation (distributed team)?
- Is the SDLC suitable for the size and complexity of software?
- Is the SDLC suitable for the type of projects we do?
- Is the SDLC suitable for our software engineering capability?



Stakeholders Need
02

business domain, stakeholders concerns and requirements, business priorities need to be considered

Decide
04

give each criterion a defined weight and score for each option. you should document this decision and share it with the related stakeholders.

Optimize
05

You can always optimize the SDLC during the project execution
You can even make your own SDLC model which optimum for your organization or the type of projects you are involved in.

BENEFITS OF SELECTING RIGHT SDLC

By Slecting the right lifecycle model, we can **improve and/or tradeoff**:

- ✓ Development speed (time to market)
- ✓ Product quality
- ✓ Project visibility
- ✓ Administrative overhead
- ✓ Risk exposure
- ✓ Customer relations, etc, etc.