Kirirom Institute of Technology

Unit 2

Process Models

Process Model Intro:

- SW Development is often a chaotic process

  Process models => Brings order and structure for the process
  Provides roadmap for SW teams

- However SE work still **“On the edge of Chaos”** (Unstable and partially structured state)

Process Model Intro Contd.

Kirirom Institute of Technology

- Operations away from equilibrium (Balanced state) => Creativity, Self organized process and increasing returns

- Absolute order => Absence of variability

- But very useful in unpredictable environments

- Model could be structured and support changes

- Too much freedom => Chaos => Problem with coherence and coordination

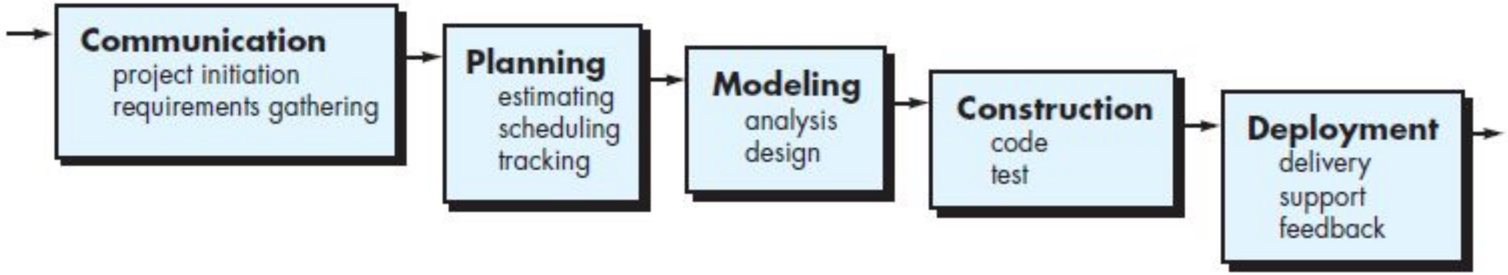
What are Process Models?

- Provides specific roadmap
- Flow of activities, actions and taskset
- Iterations
- Organization of work

Prescriptive (Predefined) process models:

- Strives for structure and order in SW development
- Provides guidelines
- They prescribe process elements (Activities, Actions and Task sets)

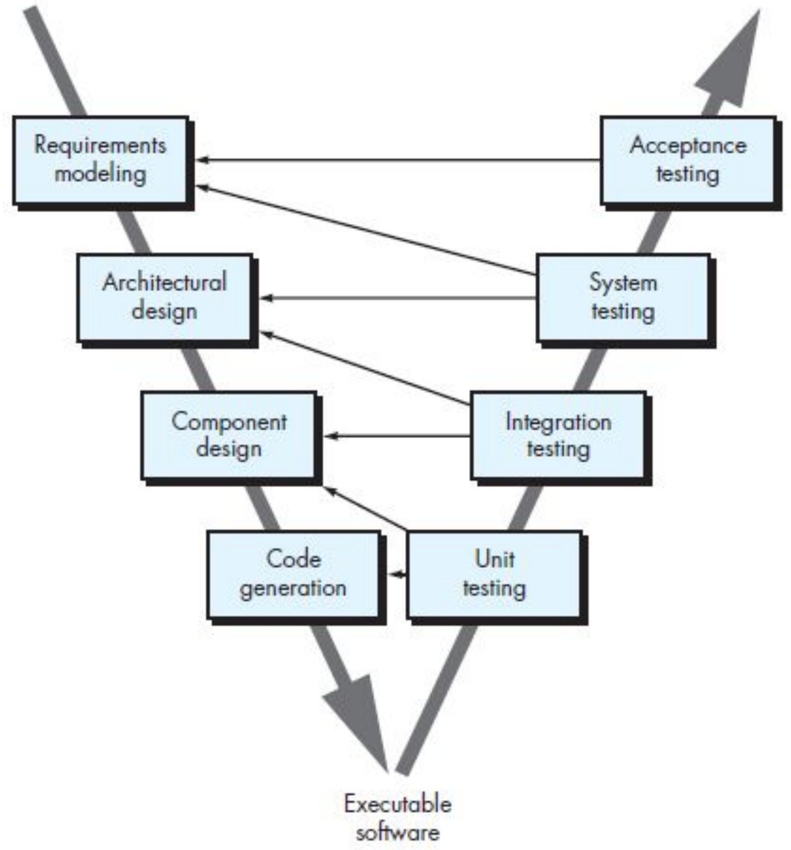
Waterfall model:



Waterfall model:

- AKA **classic life cycle**
- Systematic and Sequential approach
- Flow => Communication ... Deployment
- No way of going back to any phase
- Another variation is the **V-Model**

The V-Model


Requirements modeling
Acceptance testing
Architectural design
System testing
Component design
Integration testing
Code generation
Unit testing
Executable software

The V-Model

- depicts the relationship of quality assurance actions to the actions associated with FW activities

- Left side (FW activities) = Right side (QA Points - various tests)

- Helps visualizing how verification & validations are applied to the initial Engineering work

Criticisms on waterfall model:

Criticism 1:

- Sequential flow rarely followed by the real time projects
- Can follow iterations but indirectly, which may cause confusion

Criticisms on waterfall model:

Criticism 2:

- Waterfall model is suitable when all the requirements are readily available
- But, collecting all requirements at the beginning is difficult for clients
- Difficult for WaterFall model to accommodate the **natural uncertainty** that exists at the beginning of the project

Criticisms on waterfall model:

Kirirom Institute of Technology

Criticism 3:

- Client can get the the output only after all 5 FW activities.
- Customer need to wait until the project moves to the final stage (Deployment)
- Major blunder undetected = Disastrous

Criticisms on waterfall model:

Criticism 4:

- Linear flow leads to blocking state (One team waiting for another team to finish)

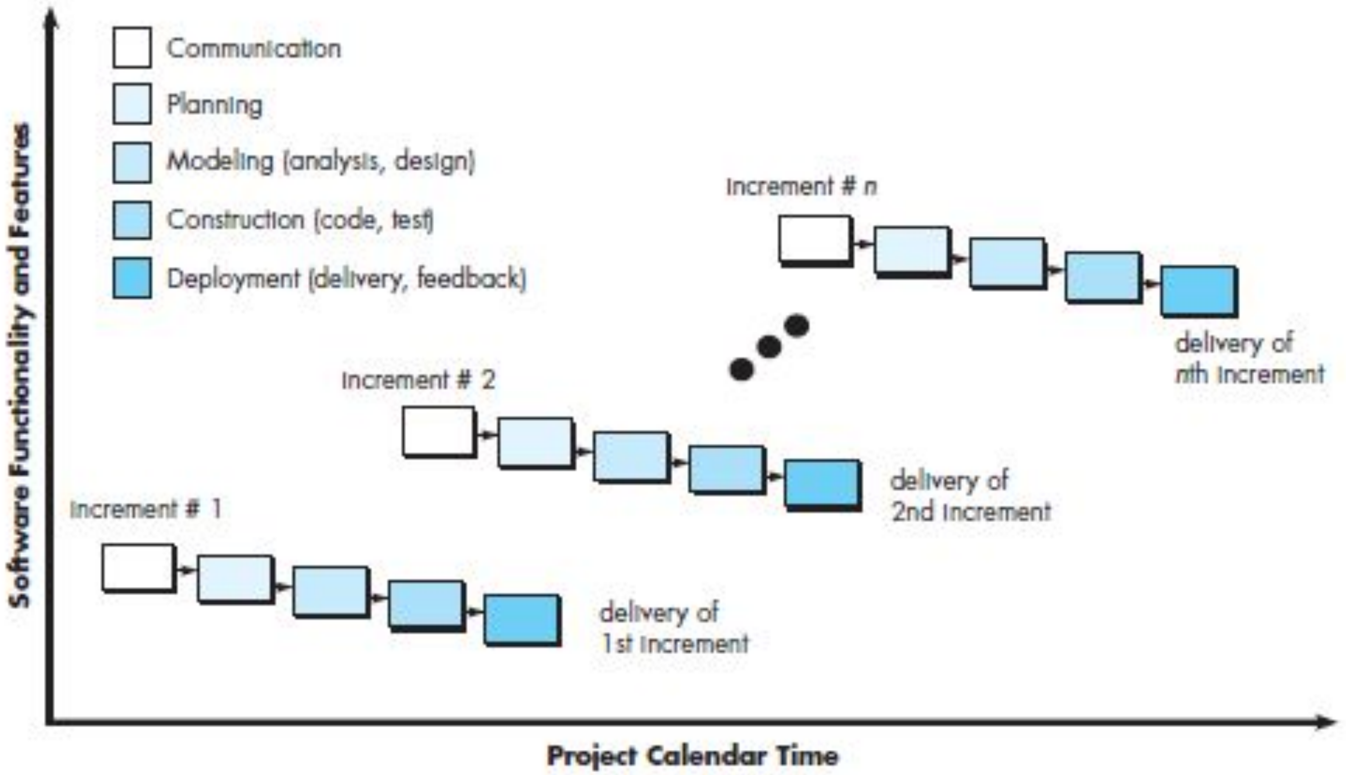
- Time spent waiting >= Time spent productive work (May be)

When to choose the Waterfall model?

- Though it has a lot of concerns, this is well suited for the projects that has **fixed requirements and fixed planning**

- Best suited for projects with Linear process flow

Incremental Process models:



Incremental Process models:

"Delivers a **series of releases**, called **increments**, that provides **progressively more functionality** for the customer as each **increment** is delivered"

- First increment is the core product
- Supplementary features are delivered in the next phases (Increments)
- Core product used by customer (Evaluation)
- Based on evaluation & requirement for the next phase the plan is developed for next increment
- Continues until the final product is produced

When to use Incremental Process models?

- Requirements are well defined
- Big projects
- Linear process flow to be followed & Keep improving the system
- Need to show limited set of SW functionality to client
- Expand the functionalities in the later releases

Evolutionary Process Model

- SW evolves over a period of time
- Business and Product requirements changes over time
- Combination of Iterative and Incremental model
- Used when only the core module of the system is known
- The additional features may be defined later
- Flexible model that accepts growth and changes over a period of time

Prototyping Model (Approach):

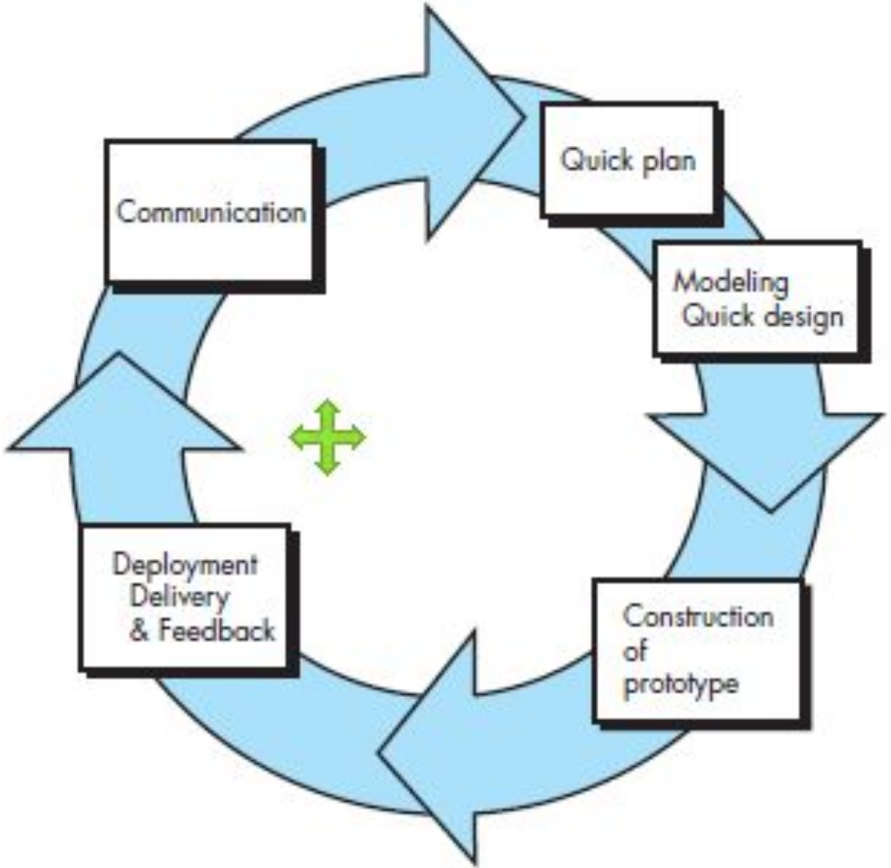
- Suitable when the client knows only objective not the clear requirement, functions & features
- Developers may be unsure: Efficiency of the algorithm, adaptability, OS compatibility
- Can also be used as a standalone process model & also used with other process models
- Used when requirements are fuzzy

How is it implemented?

Kirirom Institute of Technology

- The SW team meets the client and gets the overall objective
- Defines the overall objective, unknown requirements and where more definitions are mandatory
- Quick plan of prototyping iteration => Quick design
- Quick design => A visible aspects of SW to be shown to the client
- Quick design => construction of Prototyping
- Prototype => deployed & evaluated by stakeholders
- Feedback => Further refinement based on feedback

Prototyping Model (Approach):



Prototyping Model (Approach):

- First system - Barely usable, Slow, Too big, Awkward in use
- Redesigned after use
- Bad Prototype = The first system which can be “throw away” (Most cases)
- Good prototypes => Evolutionary => Evolves into the actual system
- Helps building something immediately and users can use the system

Issues with Prototype model

Issue 1:

- Initial prototype is created in rush to work
- Overall SW quality is not considered
- Long term maintainability is not considered
- Developers may be hesitant to throw the prototype and build a better one
- Developers may insist to use some fixes instead of the actual SW

Issues with Prototype model

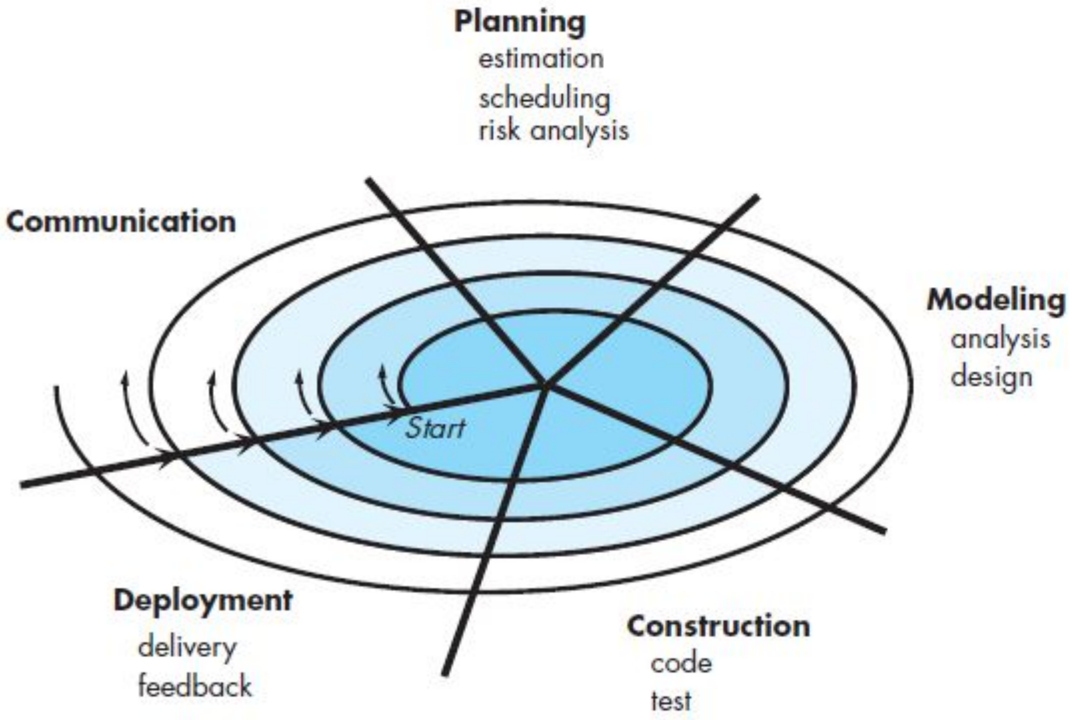
Issue 2:

- S.Engineers focus on quick prototype
- May compromise quality, long term maintainability
- Inappropriate OS, Programming language or algorithms may be used
- May have been started with poor config just to make the prototype working
- Later compromise with the quality

Spiral Model:

- An evolutionary Process Model
- Prototyping(Iterative) + Waterfall (Systematic & Sequential)
- Supports rapid development of increasingly complete versions
- **Risk Driven model** (Prioritize the risks involved, apply techniques to fix them and check if there are risks remaining)
- Two features: **Cyclic approach** and **Set of Anchor point milestones**

The Spiral Model:



1. Cyclic Approach

Incrementally…

- Growing system's degree of definition
- Decreasing its degree of Risk

2. Set of Anchor points

Ensuring

- **stakeholder commitment** to
- **feasible** and
- **Mutually satisfactory** System solutions

In Spiral Model

- SW developed in evolutionary releases
- Early releases: **Model/Prototype**
- Later releases: **More complete versions**
- Evolutionary process begins **at the centre of the spiral** with activities
- Clockwise
- Risk assessed in each evolutions and minimized with set of actions
- Anchor point milestones (Work product and conditions) are achieved in each evolutionary pass

In Spiral Model

- The 1st circuit pass => Specification
- Subsequent passes => Prototype
- After each pass => More sophisticated system obtained
- Each pass => Adjustment in project plan - cost and schedule after feedback
- Project Manager adjusts => Planned number of iterations
- Spiral model operative until the SW retired
- Can reduce risk earlier and avoid big problems in the future

Concurrent Models AKA Concurrent Engineering:

- Enables concurrent processing of Process model elements (Activities, actions & tasks)
- All the FW activities moves to various states in its life time
- All the FW activities exists (But different states)
- Example (Under development & awaiting changes states)
- This model defines a series of events (Provides accurate current state)
- Event triggers when there is a change among activity states
- Suitable for all types of development
- Events invoke actions

Issues with evolutionary Process models:

1. Project plan keeps changing due to evolution
2. If evolution too fast => Causes Chaos (Confusion among team)

   If evolution too slow => Affects productivity
3. These models focuses more on flexibility, extensibility rather than quality

- However it is possible to achieve Flexibility, extensibility and speed but a bit challenging for the Project manager to balance

Specialized Process Models:

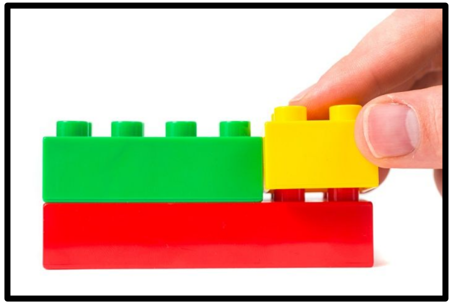
- Derived from one or many of the Traditional Process models
- Applied for narrowly defined SW Engineering approach is chosen
- Types:

    1. Component Based Development (CBD)
    2. Formal Methods Model
    3. Aspect Oriented Software Development

Component Based Development (CBD)

- Branch of SE follows Separation of Concerns (SoC)

  What is Separation Of Concern :



- Separating the Software modules based on the concern it addresses

- Reuse based approach (Creating reusable components)
- Creating Loosely Coupled Software components that constitutes the system
- Aims creating reusable SW components

How to implement CBD:

- Research available Component based products
- Create SW architecture to accommodate the components
- Component integration issues are considered
- Components are integrated into the architecture
- Comprehensive testing is done to ensure proper functionality

The Formal method:

- Encompasses set of set of activities => Mathematical specification
- Applies rigorous mathematical notations => Specify, develop and verify
- Cleanroom is another approach of Formal method
- Cleanroom focuses on error prevention than error correction
- Better for eliminating problems which are not possible with other paradigms
- Focuses on defect free SW

More suitable for...

- Engineering
- Medical
- Aerospace
- Aircraft avionics

Applications otherwise....

This may happen...

5 strategic activities of Cleanroom process

- Formal specification
- Incremental development
- Structural programming
- Static verification
- Statistical testing of the system

1. Formal specification

- SW development starts after the formal specification
- Developing stable specifications early establishes clear accountability.

Formal specifications are...

“**Mathematical specification** not based the **Natural language specification**”

2. Incremental development

- SW Project is divided into increments
- Developed and validated separately using cleanroom process

3. Structured Programming

- The SW is step wise refinement of the specification
- Limited amount of data abstracts and constructs are used

4. Static verification

- The developed SW is statically verified using rigorous SW inspections
- It is the process of checking that **the SW meets requirement** before it runs

  Eg.

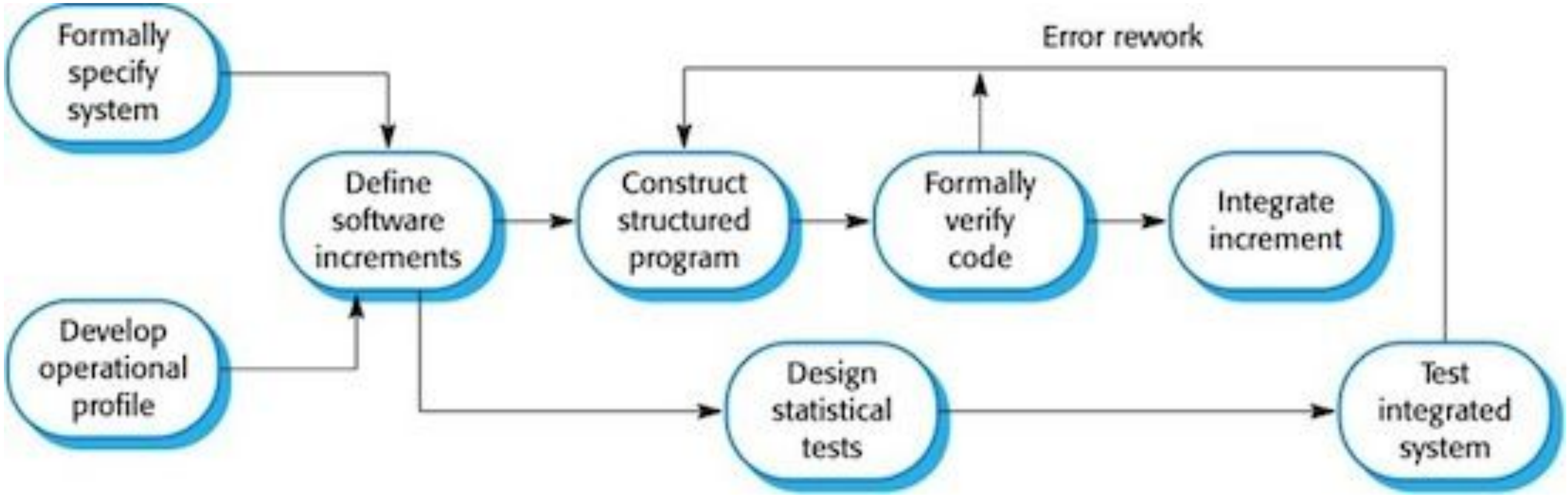
- Code conventions verification
- Bad practices (Anti pattern detection)
- SW metrics calculation
- Formal verification

- There is no Unit or Module testing

5. Statistical Testing

- The SW is tested statistically to determine its reliability
- Tested with applied mathematics

Cleanroom process



Concerns with the Formal method:

- Time consuming & Expensive
- Extensive training is required
- Difficult to use this as a model for the unsophisticated customers

But still this is most suitable and preferred for the critical systems

Aspect Oriented Software Development

- The development of Aspect Oriented Programming
- Lets understand what is Aspect Oriented Programming first

The different paradigms:

Kirirom Institute of Technology

- Procedure Oriented Programming - Program into different modules
- Object Oriented Programming - Emphasize on class & Objects
- Functional Programming - Emphasize functional suitable for scientific apps
- Aspect Oriented Programming - Emphasizes aspects

What are aspects?

- Specialized class that addresses any of the crosscutting concerns
- Crosscutting concerns are applied all over the project

What are cross cutting concerns?

- Security
- Profiling
- Logging
- Transaction Management

=> **These are also called as non functional requirements**

Let’s see an illustration to understand it better

Benefits of AOP

- Address Cross cutting concerns
- Reusability
- Faster development
- Focus on one aspect and develop
- Enabling / Disabling aspects is easier at runtime

The Unified Process:

- Takes the best features from the all Process model
- But characterize to implement the best principles of Agile
- Suitable for complex and big SW design
- Recognizes customer communication & streamlined methods

Characteristics of UP:

- Emphasizes important role of SW architecture
- Helps architect focus on the right goals

Right goals:

- Understandability
- Reliance to future goals
- Suggests iterative and incremental process flow

The UML

- Was the result of the combination of **“The Object Oriented Analysis”**
- UML - **“Unified Modeling Language”**
- Used for the development of **“Object Oriented Systems”**

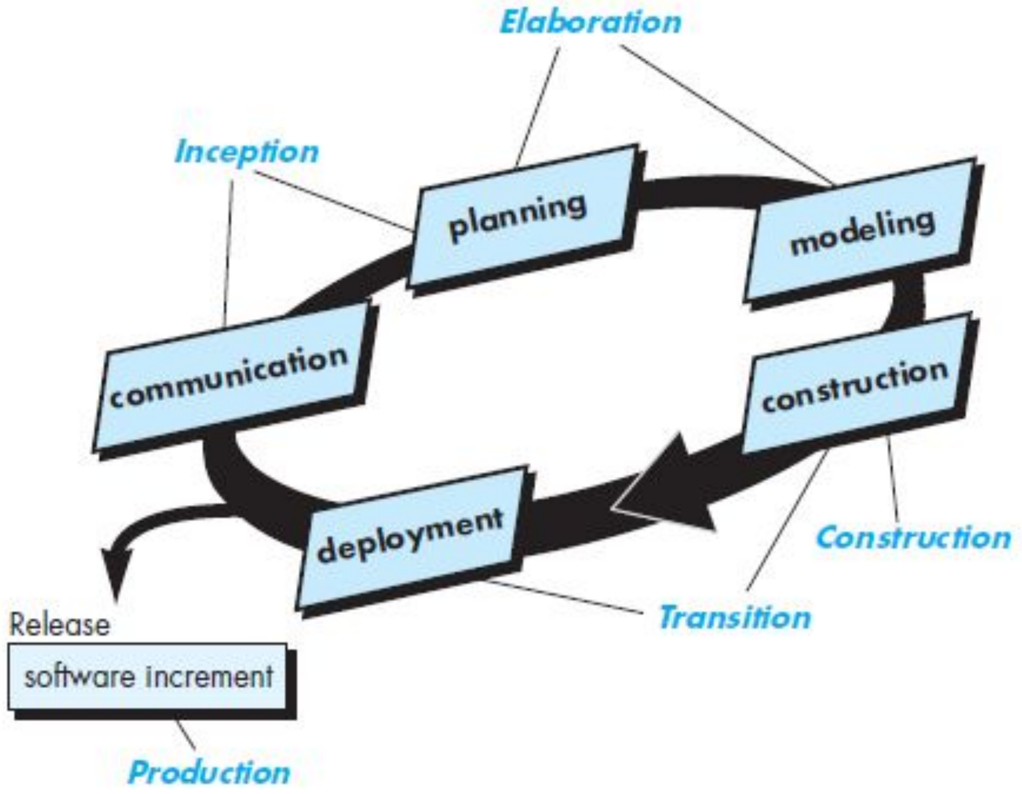
(Will be discussed in the later units in detail)

Different phases of Unified Process:

- Inception
- Elaboration
- Construction
- Transition
- Production

(UP maps the above phases with the generic process model)

The Unified Process



Inception (Communication - Planning):

- Combination of **Communication** and **Planning**
- Fundamental business Requirements are identified (Discussion - Stakeholder)
- Rough architecture & iterative project plan is developed
- Architecture - Tentative outline of the major sub system & Functions & features to achieve them
- Defines resources, risks involved and establishes basis for the phases

Elaboration (Planning - Modeling)

- Refines use cases, expands architectural representation
- Includes 5 views of SW

1. Use case model - Defines interactions between actors
2. Analysis model - Technical representation of the system
3. Design model - Abstraction of the implementation of the system
4. Implementation model - Defines the components / artifacts of the system
5. Deployment model - Defining nodes (Physical architecture of the system)

Construction

- The analysis & the plan made in elaboration are completed in this phase
- The final version of each SW increment is released
- The designed components are implemented with Source code
- Unit test are designed
- Components are assembled (Integration)
- Integration tests are conducted

Transition (Construction - Deployment):

- Later stages of the construction activity
- First part of the next activity (Deployment)
- Beta testing (SW given to users)
- Delivery & feedback
- Documentation - User manuals, troubleshooting guides, installation procedures

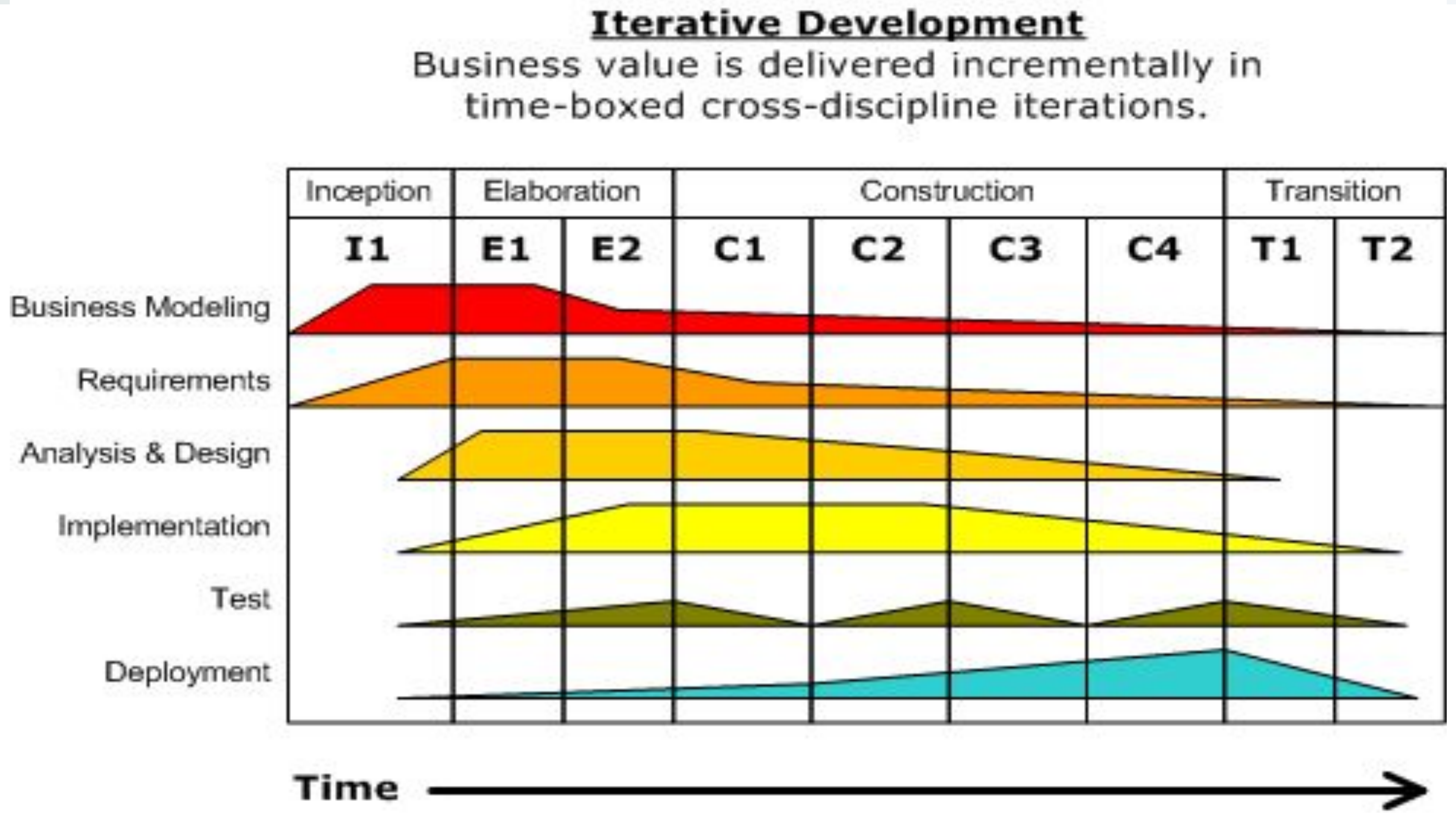
Production

- Coincides with deployment
- SW is monitored
- Defect reports are made
- Change requests are submitted & evaluated

The Unified Process:

- UP phases are not occurred in sequence
- Can happen concurrently
- Workflow is distributed across all UP phases
- Workflow identifies Taskset

Sample workflow of Unified Process:



Personal & Team process Models:

- Best SW process = Close to people using it
- Broader level process = Organizational level process
- Broader level process effective only if it is customized to the SW dev team
- Personal SW Process (PSP) + Team SW Process (TSP) can be effective

Personal process Models:

- Each individual may follow his/her own model to finish the tasks which
- may be incomplete or partially complete and change on daily basis
- Mostly ineffective, inefficient and unsuccessful
- Watts Humphrey recommends 5 phases to apply called PSP
- Measures the work product & resultant quality
- Practitioner responsible for project planning
- Empowers practitioners control quality of the product

Personal Process Models

- Planning
- High level design
- High level design review
- Development
- Postmortem

Planning:

- Resource estimation
- Defect estimation
- Record all metrics in spreadsheet / template
- Identify development tasks
- Project schedule creation

High level design:

- Component design (with extra specs) created
- Prototypes are built if there are uncertainty
- Issue recording & tracking

High level design review:

- Design is verified with formal verification method to uncover errors
- Metrics are maintained for important tasks

Development:

- Component level design is refined & reviewed
- Code generation, review, compile & testing is done
- Metrics are maintained for important tasks

Postmortem:

- Measures & metrics are collected
- Analyzing the effectiveness of the process
- Modified based on Measures & metrics

Pros of PSP:

- Identify errors & types of errors early
- Rigorous assessment activity on all work products produced
- Disciplined metrics based approach
- PSP can be used partially to improve the personal process

Cons of PSP:

- Have more to do with human nature & organizational inertia (Resistance)
- Culture shock for many practitioners
- But proved to improve quality and productivity
- Intellectually challenging to practitioners because high level of commitment needed

Objectives of Team Process Model (TSP):

- Goal is **to create self directed project team** that **organize itself to produce HQ SW**

  What is a Self Directed team?

  Is a team that

  -Plan & track their work

  -Establish goals

  -Own their process plans

  -It may be a pure SW team or Integrated Product Teams(IPT)

Objectives of TSP Contd.

- Show Managers how to coach & motivate team & help them to sustain peak performance
- Accelerating SW Process improvement & complying with CMMI 5
- Provides improvement guidance to high maturity organizations

AGILE Development

What is Agile?

You would hear any of the following definitions:

- Iterative approach
- Deliver faster in less time
- Ability to create quickly & quickly respond to change
- Philosophy to rapidly deploy an app
- Fully loaded toolkit
- Just a mindset of thinking

Kirirom Institute of Technology

Well all of them are correct but not the official definitions

What is Agile?

- Agile practices approach discovering requirements and developing solutions through the collaborative effort of self-organizing and cross-functional teams and their customer(s)/end user(s)
- Advocates adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages flexible responses to change

BC
Middle Ages
century
Ancient Greece
ROME
XIX
HISTORY
WORLD WAR
Herodotus
XX
Periods
dreamstime

Agile History:

- 17 gentlemen joins together in Utah USA - Feb - 2001
- Enjoyed eating, skiing and also discussed the problems in the SW industry
- Found the agile methodology
- Agile Alliance was formed

Agile

Works based on 4 values and 12 principles

The 4 values of Agile Manifesto:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

http://agilemanifesto.org/

12 Principles of Agile

Kirirom Institute of Technology

Customer Satisfaction

Welcome Change

Deliver Frequently

Working Together

Motivated Team

Face-to-Face

Working Software

Constant Pace

Good Design

Simplicity

Self Organization

Reflect and Adjust

1. Customer satisfaction

Our **highest priority** is to **satisfy the customer** through **early and continuous delivery of valuable software**

2. Welcome change

Welcome changing requirements, even late in development. Agile processes harness change **for the customer's competitive advantage**.

3. Deliver frequently

Deliver working software frequently, from a couple of weeks to a couple of months, with a **preference to the shorter timescale**.

4. Working together

Business people and developers must **work together daily throughout** the project.

5. Motivated Team

Build projects around **motivated individuals**. Give them the **environment and support they need**, and **trust them** to get the **job done**.

6. F2F conversation

The most **efficient and effective method of conveying information** to and within a development team is **face-to-face conversation**

7. Working Software

Working software is the **primary measure of progress**.

8. Constant pace

Agile processes promote sustainable development.The **sponsors, developers, and users** should be able to **maintain a constant pace indefinitely**.

9. Good Design

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. Self organization

The **best architectures, requirements, and designs** emerge from **self-organizing teams**.

12. Reflect & adjust

At **regular intervals**, the **team reflects** on how to become **more effective**, then **tunes and adjusts its behavior** accordingly

The Agile Test:

1. Does what we're doing at this moment support the early and continuous delivery of valuable SW?

2. Does our process welcome and take advantage of change?

3. Does our process lead to and support the delivery of working functionality? Are the developers and the product owner working together daily?

4. Are customers and business stakeholders working closely with the project team?

5. Does our environment give the development team the support it needs to get the job done?

6. Are we communicating face to face more than through phone and email?

Agile Test Contd..

7. Are we measuring progress by the amount of working functionality produced?

8. Can we maintain this pace indefinitely?

9. Do we support technical excellence and good design that allows for future changes?

10. Are we maximizing the amount of work not done - namely, doing as little as necessary to fulfill the goal of the project?

11. Is this development team self - organizing and self - managing? Does it have the freedom to succeed?

12. Are we reflecting at regular intervals and adjusting our behavior accordingly?

What Agile is not...

1. A specific way of doing software Engineering
2. A framework, standard or a process
3. And end-goal by itself (But a journey towards continuous improvement)
4. An excuse to stop / reducing documentation or an opportunity to eliminate planning
5. One size fits all (Organization's vision, culture & specific needs)
6. scrum - Scrum applies Agile principles
7. Kanban - Scrum applies Agile principles
8. Limited to SW development

Agile overview Summary

1. Refers to any process that aligns with the concepts of Agile manifesto
2. Based on incremental delivery and iterative approach
3. Encourages constant feedback from the end users
4. Deliver value faster and foster the ability to better respond to market trends

Waterfall vs Agile difference

Waterfall model

1. Built in phases
2. Painful to revisit previous phases
3. High dependency on critical paths (Blocking issues)
4. Customer can't interact with product until its fully complete

Agile

1. Iterative approach
2. Regular feedback intervals
3. Iterations resolve blocking issues & let you interact with the product
4. Quickly adapt new requirements/changes
5. Shared skill set among the team

Kirirom Institute of Technology

When to choose what?

Use Waterfall if...

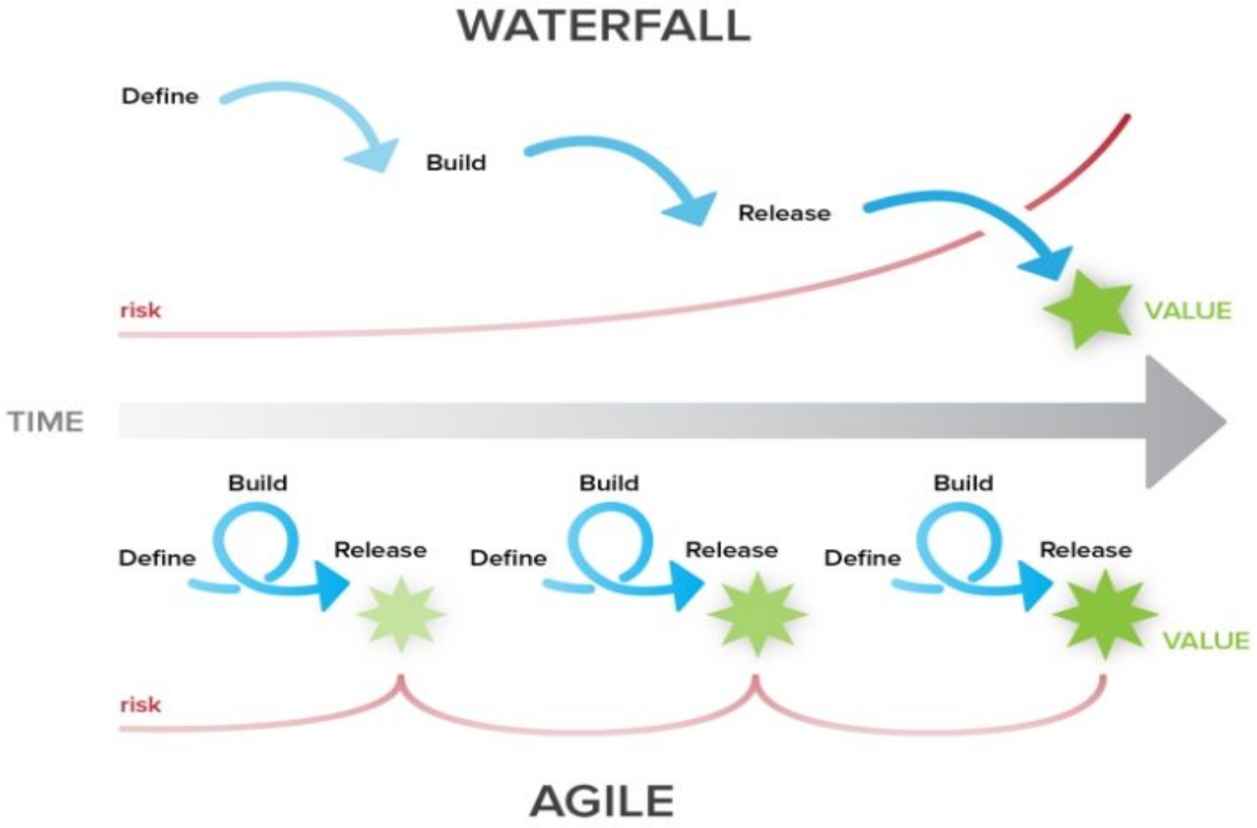
1. You don't expect changes in scope & fixed price contracts
2. Simple projects or you have done it many times before
3. Requirements are very well known and fixed
4. Customers know exactly what they want in advance
5. Orderly & predictable projects

Use Agile if...

1. Final product isn't clearly defined
2. Clients/stakeholders need the ability to modify the scope
3. Anticipate any kind of changes during the project
4. Rapid deployment is the goal

Agile vs Waterfall model

Kirirom Institute of Technology



Agile Advantages and Disadvantages

Agile advantages:

- Change is embraced (Planning, refining backlock)
- End-goal can be unknown
- Faster, high-quality delivery
- Strong team interaction (Frequent communication and F2F interactions)
- Customers are always heard (To see work being delivered, Share input and have an impact on the end product)
- Continuous improvement (Feedback encouraged from users and team members)

Agile Disadvantages:

- Planning can be less concrete
- Team must be knowledgeable
- Time commitment from developers is required
- Documentation can be neglected
- Can fail due to unrealistic expectations

Agile Key concepts

User Stories:

- Work divided into functional increments called user stories

- Short description of a feature from the perspective of the person who desires the new capability

- User story should be potentially shippable

- Size of user story is measured in story points

Stories are:

- Role (Who) + Action/Goal (What) + Benefit (Why) and acceptance criteria

Role (Who):

- User (Should be an actual human who interacts with the system)
- Be as specific as possible
- The development team is not a user

Goal/Action (What):

- The behavior of the system must be written as an action
- Usually unique for each user story
- Describe intent and not the technical feature
- The system implied doesn't get written in the story
- Use active voice, Not passive voice (I can be notified)

Benefit (Why):

- The benefit should be a real world result that is non-functional or external to the system

- Many stories may share the same benefit statement

- The benefit may be for other users or customers, not just for the user in the story

User stories example:

- As a **Customer**, I want **Shopping cart feature** so that **I can easily purchase items online**.
- As a **Manager**, I want to **generate a report** so that **I can understand which dept needs more resources**
- As a **customer**, I want to **receive an SMS when the item is arrived** so that **I can go pick it up right away**
- As a **manager**, I want to **be able to understand my colleagues progress**, so that **I can better report our success and failures**

Acceptance criteria:

Are the agreed upon measures to prove you have done.

Acceptance criteria:

- Set of predefined requirements that must be met in order to consider a story is completed

- The definition of done (To consider the story as done)

- Crucial for the user story for unit test, Quality Assurance and story acceptance by product owner or stakeholder

Things to consider while writing Acceptance Criteria:

- Should be testable

- Straight forward must provide yes/no result

- Clear and concise

- Everyone must understand the criteria

- Should provide user perspective (Real user experience)

How not to write?

- Ambiguous language

- Subjective / Judgemental language (Better, good, allowable)

- Generalization (All the time, never, everyone, always)

- Observed behavior that is currently not directly related to the story currently under consideration

Acceptance criteria must explain:

What is expected and not **How to implement**

Who is responsible for writing acceptance criteria?

- Anyone in a cross functional with the consent of Product owner / Business Analyst
- Most preferably the PO / BA

When should be written?

- The user story and acceptance criteria must be written before the development begins

How to format user story acceptance criteria?

- There is no right or wrong way to write acceptance criteria.
- Trial and error

Why is it important to write Acceptance criteria?

- **Define boundaries** for a user story

- Help product owner answer **what he/she needs** in order for this feature to **provide value** (Minimum functional requirements)

- Help them **gain a shared understanding of the story before development** has started

- Help developers and testers **to derive tests**

- Help developers **to know when to stop adding more functionalities** to a story

INVEST:

I - Independent (Of all others) - No inherent dependency on another user story

N - Negotiable (Not a specific strict contract always space for negotiations)

V - Valuable Must deliver value to stakeholder / Business

E - Estimable (Must be able to estimate the user story)

S - Story shouldn't be big. Must be small and achievable

T - Testable (In principle, even if there isn't a test for it yet)

Example of a good user story & Acceptance criteria:

User Story:

- As an **online visitor**, I want to **add products in my shopping cart** so that **I can purchase multiple products at one go**

Example of a good user story & Acceptance criteria:

Acceptance criteria:

- Products can be added to the cart
- Products can be removed from the cart
- Shopping cart will be empty initially
- Shopping cart will be empty after purchase
- Products can be added with multiple quantities in the cart
- Shopping cart will show the total product breakdown quantity and cost with grand total

EPICS:

- Are series of user stories with a broader strategic objective
- Is a large user story that can’t delivered in a single iteration
- It is split into smaller stories
- There are no standards available to write Epics
- Epics can be written like stories

Epic example:

1. As a **VP internship** I want to **see the KIT point status of any KIT student** so that **I can see the progress of the student**

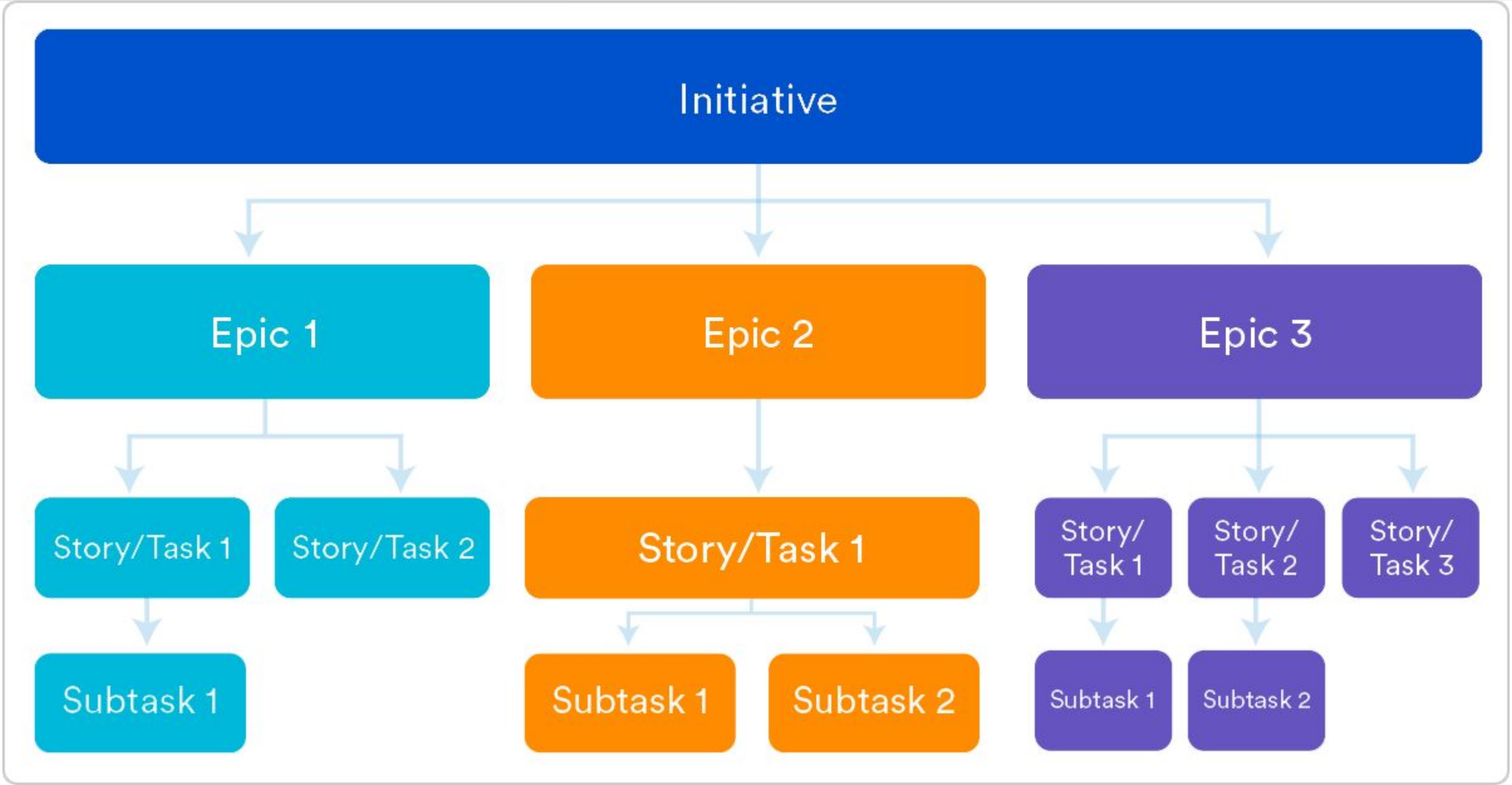
1.a. As a **VP internship** I want to **see the yearwise KIT point status of any KIT student** so that **I can see the progress of any student by year**

1.b. As a **VP internship** I want to **see the projectwise KIT point status of any KIT student** so that **I can see the progress of any student by project**

Initiatives:

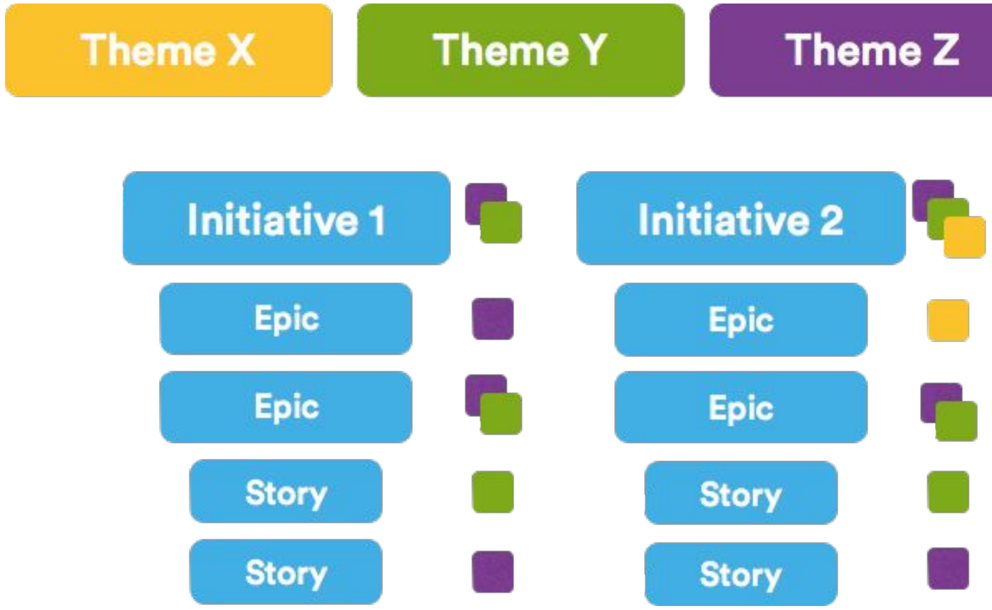
- Collection of epics that drive towards a common goal
- Product roadmap is a collection of initiatives plotted along timeline
- Completion of epics will lead to the completion of initiatives
- The next picture will give you the idea about the relationship between initiatives, epics and stories

Initiatives, epics and stories



Themes:

- Are larger focus areas that span across the organization
- Themes leads to the creation of initiatives and epics
- The relationship between themes and initiatives can be Many to Many



Themes, Initiatives, Epics and story framework

- This Framework allows for more strategically sound decisions
- Story points are the fundamental units of measure

Benefits of story points:

- Better informed decisions
- Performance monitoring
- Timeline estimation
- Can find the balance between too much work and too little work

Case study:

Kirirom Institute of Technology

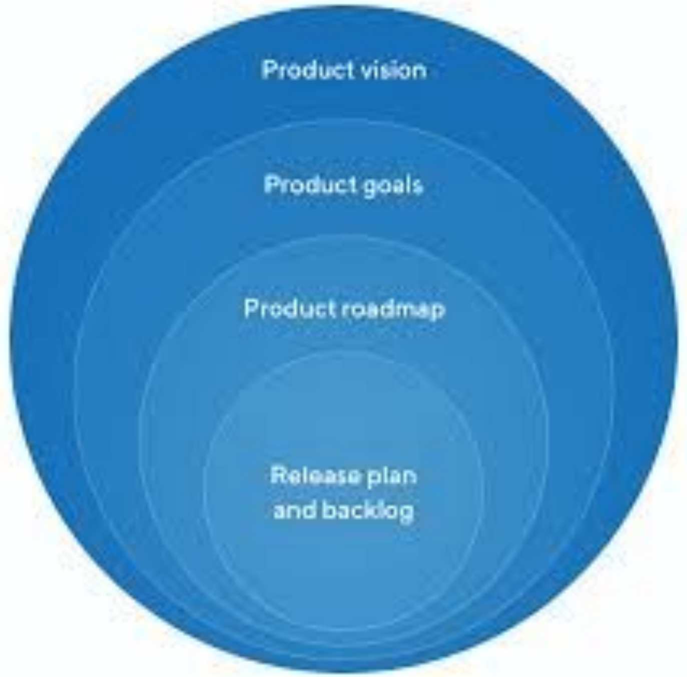
- Theme: KIT wants to be a leading innovative university in Asia
- Initiatives: Internship model, Good curriculum, Job placement
- Lets write the epics and stories for the initiative one

Product Backlog:

- Task level todo list for the product
- Lists and prioritizes the tasks
- Consists of stories, new features, changes required, bug fixes etc
- Single source of requirement that defines the product
- Live and get updated (Add, remove, update etc)
- Contains short description of of functionality desired in the task
- Can be physical or electronic form
- Owned and maintained by Product owner

Product Roadmap vs Product backlog

- Roadmap is a broader strategy to complete the product
- Backlog contains the tactical details of development to follow the roadmap



Burndown charts

- Are helpful to update the daily progress
- Helps identify the progress flow
- Good for estimation
- Timely prevention of issues
- Good for team’s motivation

Creating Burndown charts

Leak was asked to create a burndown chart for a project by her leadership to know how things are progressing based on what was planned

Note Project has 10 tasks to be finished in 10 days

Velocity:

- At the end of each iteration the team adds up effort estimates associated with user stories that were completed during that iteration. The total is called velocity

- Velocity = units of work completed in a given timeframe

- Units of work = Hours/ user stories / story points
- Time frame = Iterations / sprints / weeks

MVP - Minimum Viable Product

- It is the version of the new product that allows a team to collect the **maximum amount of validated learning** about customers with **the least amount of efforts** - *Agile Alliance*

What is MVP?

- First version of the product (solution of a problem)
- Contains enough features with high enough quality
- To attract the first set of customers
- To be able to gather the valuable feedback from customer (How they use the product)
- The term coined by Frank Robinson in 2001

“Core purpose - To gather data and feedback from customer”

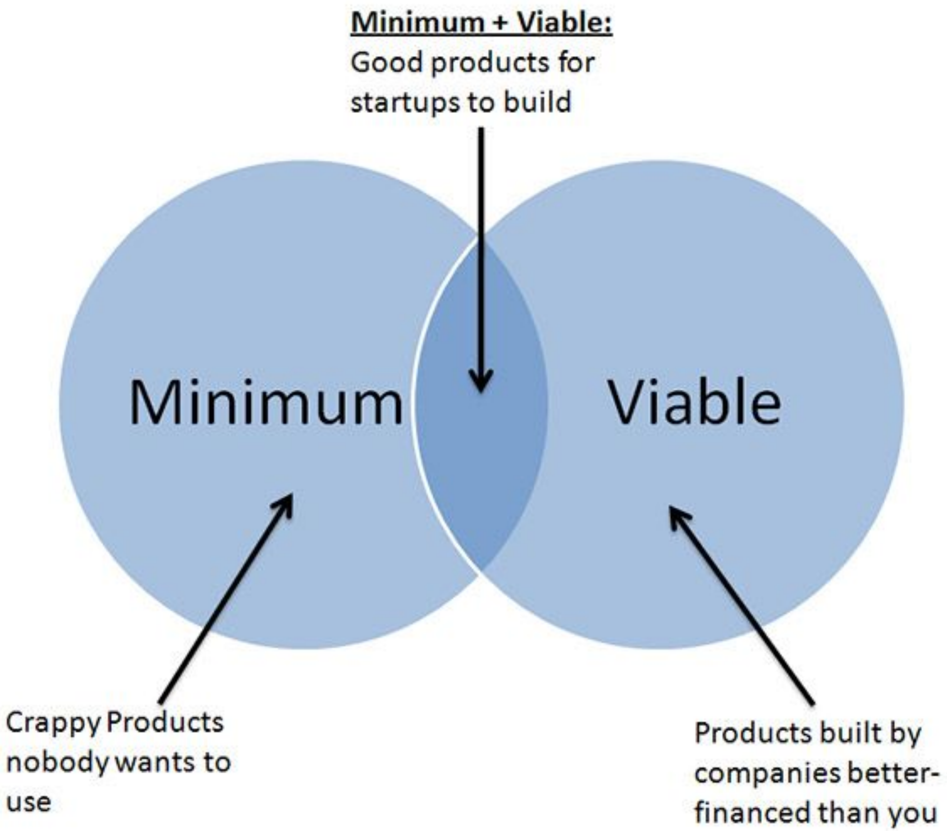
The Idea:

- It is not about producing something with Minimum functionality

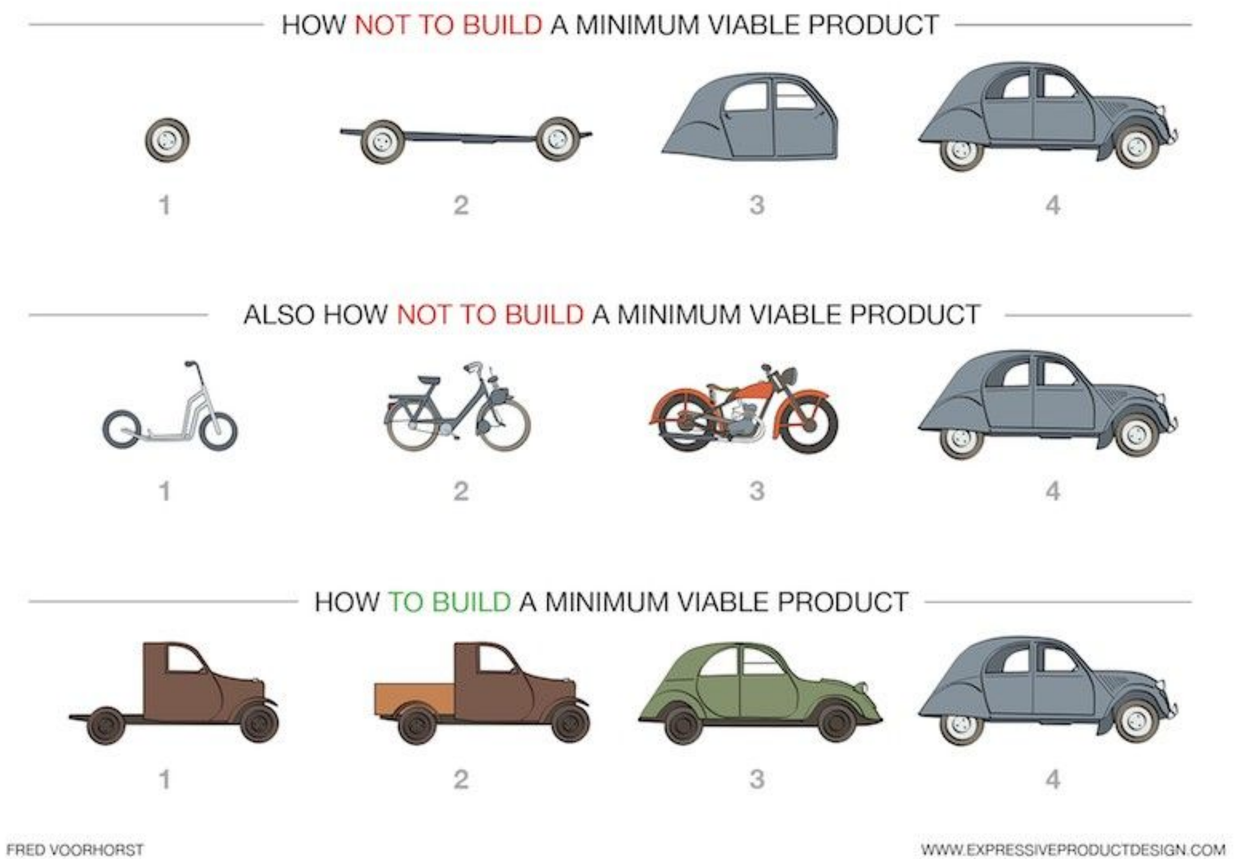
“It is about building something **with Minimum functionality** and **with quality** that allows you **to learn** about **your product, customer and what you need to do**

“RISK MANAGEMENT”

Minimum Viable Product


Minimum + Viable:
Good products for startups to build
Minimum
Viable
Crappy Products nobody wants to use
Products built by companies better-financed than you

How to Build an MVP?

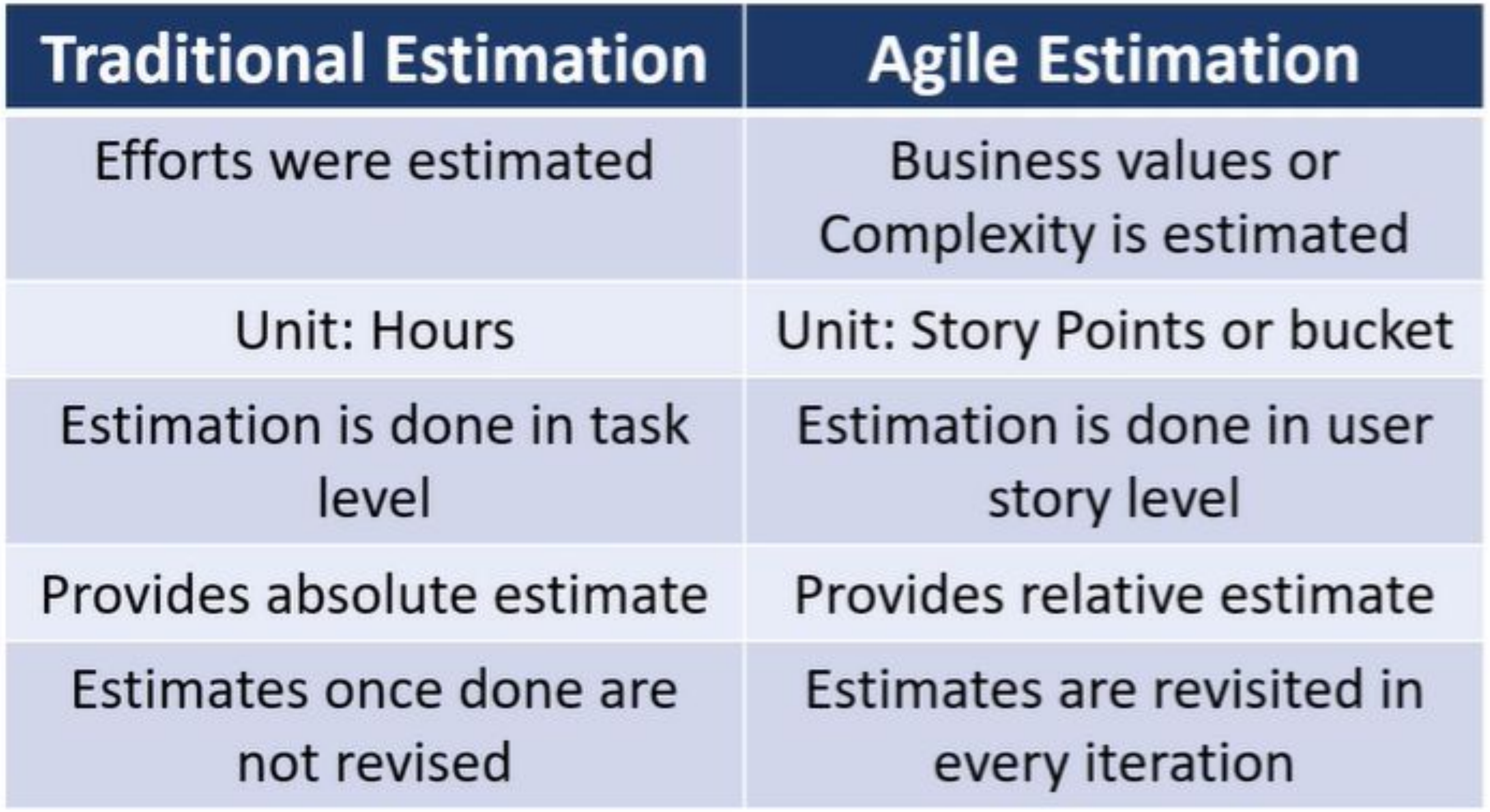

HOW NOT TO BUILD A MINIMUM VIABLE PRODUCT
1
2
3
4
ALSO HOW NOT TO BUILD A MINIMUM VIABLE PRODUCT
1
2
3
4
HOW TO BUILD A MINIMUM VIABLE PRODUCT
1
2
3
4
FRED VOORHORST
WWW.EXPRESSIVEPRODUCTDESIGN.COM

Agile Estimation:

- Agile follows Relative estimation
- Traditional models follows absolute estimation

What is relative estimation?

Agile Estimation:



Traditional Estimation	Agile Estimation
Efforts were estimated	Business values or Complexity is estimated
Unit: Hours	Unit: Story Points or bucket
Estimation is done in task level	Estimation is done in user story level
Provides absolute estimate	Provides relative estimate
Estimates once done are not revised	Estimates are revisited in every iteration

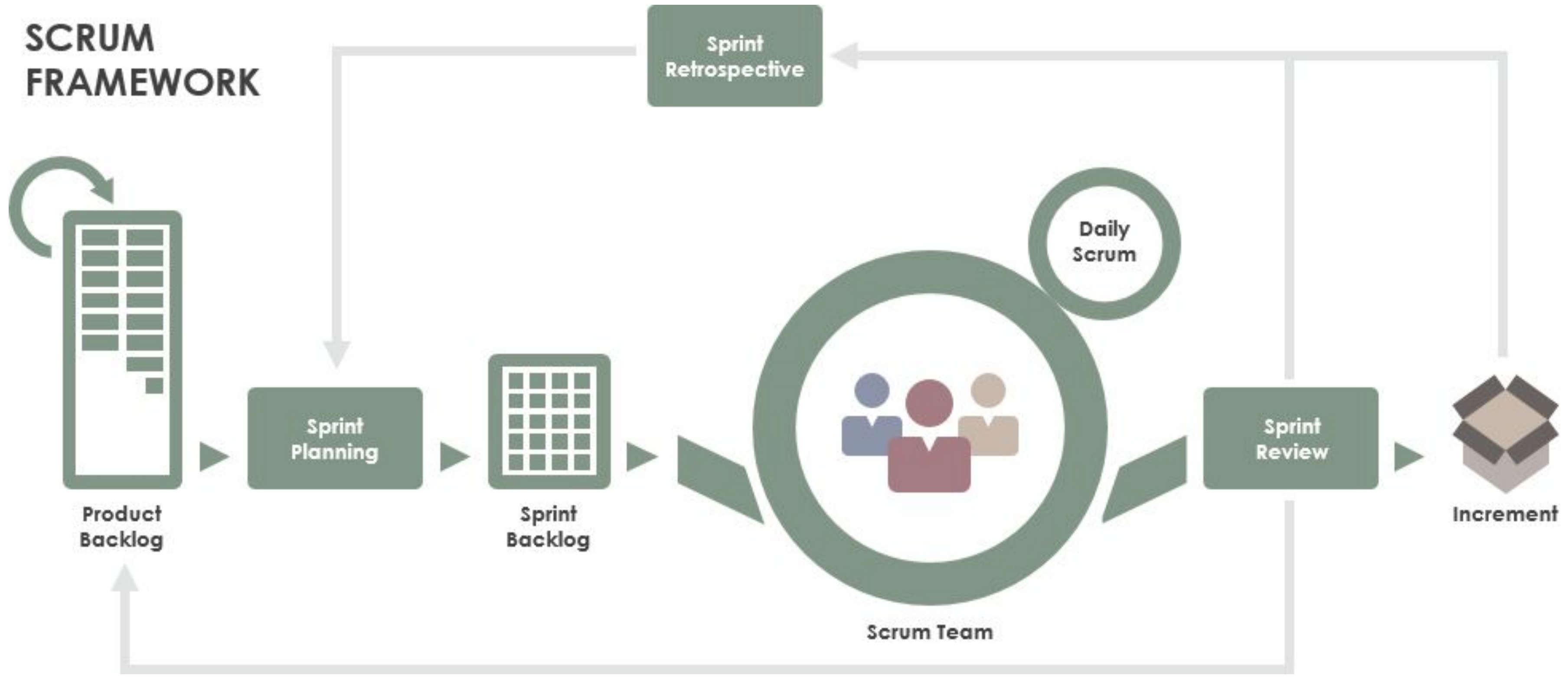
Scrum Framework:

- Scrum is a process framework used to manage product development and other knowledge work

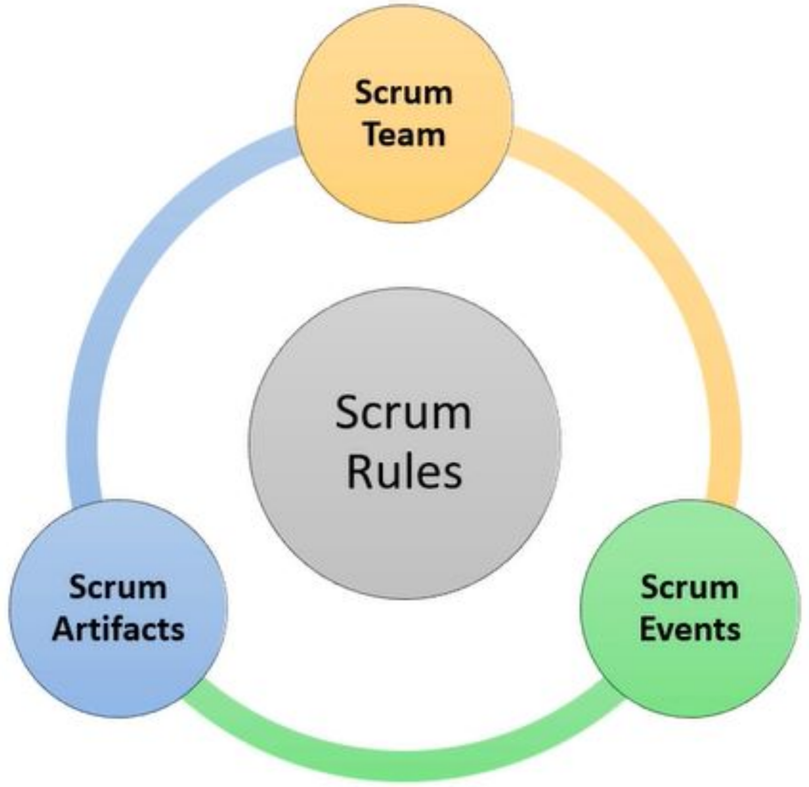
  - Agile Alliance

- Scrum is a set of rules for structuring the team, processes and techniques for making the development process agile..

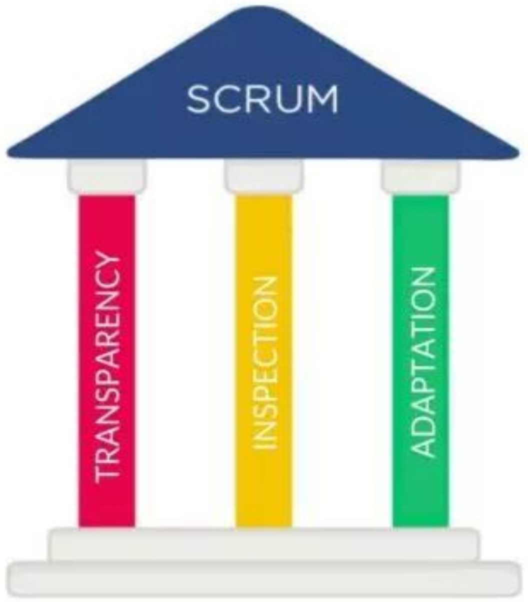
The Scrum flow:



Scrum components:



The 3 pillars of Scrum:



Pillar 1: Transparency:

- Required information is available and flows properly
- Can also be called as clarity on what is to be achieved
- High transparency between dev team and other stakeholders
- Common definition of task completion between stakeholders

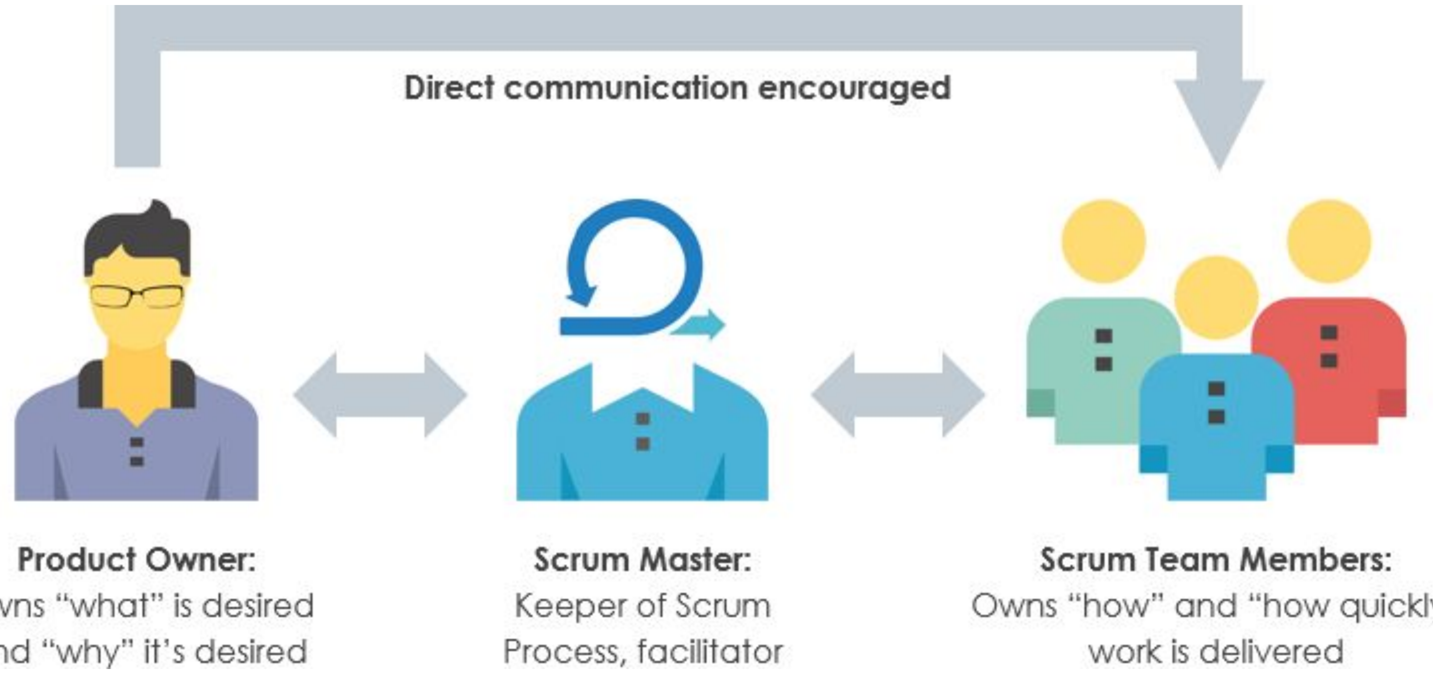
Pillar 2: Inspection

- Regular monitoring to early detection of issues
- To identify what is achieved and what is pending
- Scrum events and artifacts are aimed at achieving this
- Identify if there is any deviation in the intended final product

Pillar 3: Adaptation

- Course correction to prevent further deviation
- Preventive measures to prevent deviation

The scrum team



Product owner:

- Knows **what and why a task need to be done**
- **Must be a single person** not a committee
- Maintains a master **list of pending tasks** AKA Product Backlog
- Maintains the **priority of tasks** (Will be visible to the dev team)
- Makes sure the **dev team understands the tasks properly**
- Changes made to the Product Backlog **must go via Product Owner**

Development team:

- **Group of people** works on the requirement
- **Delivers a potential releasable product at the end of each sprint**
- **How the tasks must be done** must be given to the **dev team**
- **Cross functional team** and **no external dependency**
- It is **self organizing**
- There will be **no titles and no subteam** within a scrum dev team
- Team size: **3 to 9 is good** but depends on the various other aspects as well

Scrum Master:

- Person who masters scrum framework
- Focuses on improving the team’s efficiency
- Solves disputes between scrum team and PO
- Makes sure proper implementation of scrum process
- Removes impediments (Hindrances) from scrum team
- Arranges daily standup, retrospective and sprint planning meetings
- Helps PO to create a good product backlog
- Plan scrum events and artifacts (Will be discussed later)

Scrum events:

- Events can be monthly/weekly/daily
- Time boxed (There is a limited duration of these events)
- Intended to improve efficiency (Improve productivity)
- Any event seem counter productive then, it must be reported to the scrum master
- Must be created based on the three pillars of scrum

Sprint:

- Instead of giving a task based timeline
- The team is committed to release an increment after x days of work
- Based on the timeline the tasks are adjusted
- The increment must be releasable after each sprint
- The spring timeline must be less than or equal to 30 days maximum
- The longer the duration the more the complexity

Sprint Planning:

- Happens once for each sprint
- Plan for the sprint is fixed in this planning meeting
- Participants: All members of the scrum team
- Time boxed: 2 hours for a sprint of one week (Max: 8 hours)
- Two agendas:

I. **What** is to be done?

II. **How** it will be done?

Daily Scrum:

- Attended mainly by the Dev team
- Facilitated by Scrum master
- PO may attend but not necessary
- Agenda:

  1. Regular inspection
  2. Early adaptation

- 15 Minutes a day
- What has been completed and what will be done

Sprint Review:

Kirirom Institute of Technology

- Attendees: Entire scrum team & other stakeholders
- Objective: To Improve efficiency
- Increase transparency among the scrum team and stakeholders
- Time Box: 4 Hours

Sprint review meeting:

- Agenda: Review of the Product Increment
- PO invites the team member & other stakeholders
- Highlights what has been completed and what not completed
- Dev team demonstrate the completed items
- Dev team shares experience, highlight challenges
- Stakeholders may ask questions
- PO opens the backlog & discuss on what to do next
- Update Backlog based on latest priority

Sprint retrospective:

- Agenda: To discuss the performance of the team
- What went well?
- What went wrong?
- Not only to improve the output instead focus on the team’s growth also
- Time box: 3 Hours or less
- At the end of this event the team must know:

  1. Area of improvement
  2. Plan of working on them

Scrum artifacts:

- Product backlog
- Sprint backlog
- Increment

Increment:

- The final releasable product after each sprint
- It should be usable
- It should be inspectable
- It should include increments of previous sprint
- As the team gets matured the “Definition of Done” becomes stricter