Foundation for Advancement of Science & Technology

Agile Development :-

Large Projects

Small Chunks (Iterations)

Release → Feedback → Enhance → re-release

Advantages
- Frequent delivery
- Face to face communication with clients.
- Changes
- Time

Disadvantages
- Less documentation
- Maintenance Problem.

Agile development Techniques :-

- Extreme Programming :-

Select user stories for this release → Break down stories to tasks → Plan release → Develop integrate test software → release software → Evaluate System

- User stories for requirements
* In XP, a customer or user is part of the XP team and is responsible for making decisions on requirements.
* User requirements are expressed as user stories or scenarios.
* These are written on cards and the development team break them down into implementation tasks. These tasks are basis of schedule and cost estimates.
* The customer chooses the stories for inclusion in the next release based on their priorities and the schedule estimates.

Examples of refactoring :-

* Re-organization of a class hierarchy to remove duplicate code.
* Tidying up and renaming attributes and methods to make them easier to understand
* The replacement of inline code with calls to methods that have been included in a program library.

Foundation for Advancement of Science & Technology

Agile Development :-

Large Projects

↓

Small Chunks (Iterations)

↓

Release → re-release

↓

Feedback → Enhance

Advantages

- Frequent delivery
- Face to face communication with clients.
- Changes
- Time

Disadvantages

- Less documentation
- Maintenance Problem.

Agile development Techniques :-

- User stories for requirements
- In XP, a customer or user is part of the XP team and is responsible for making decisions on requirements.
- User requirements are expressed as user stories or scenarios.
- These are written on cards and the development team break them down into implementation tasks. These tasks are basis of schedule and cost estimates.
- The customer chooses the stories for inclusion in the next release based on their priorities and the schedule estimates.

- Extreme Programming :-

Select user stories for this release → Break down stories to tasks → Plan release

↓

Develop integrate test software → release software → Evaluate System

Examples of refactoring :-

- Re-organization of a class hierarchy to remove duplicate code.
- Tidying up and renaming attributes and methods to make them easier to understand
- The replacement of inline code with calls to methods that have been included in a program library.

Foundation for Advancement of Science & Technology

Test-driven development :-

* Writing tests before code clarifies the requirements to be implemented.
* Tests are written as programs rather than data so that they can be executed automatically. The test includes a check that it has executed correctly.
  - Usually relies on a testing framework such as Junit.
* All previous and new tests are run automatically when new functionality is added, thus checking that the new functionality has not introduced errors.

pair programming :-

* In pair programming, programmers sit together at the same computer to develop the software.
* Pairs are created dynamically so that all team members work with each other during the development process
* Sharing knowledge → reduce overall risk to project when member leaves.
* More efficient than 2 programmers working separately.
* Common owner of code
* Informal review process (reviewed by more than 1 person)
* Encourages refactoring.

- Scrum :-

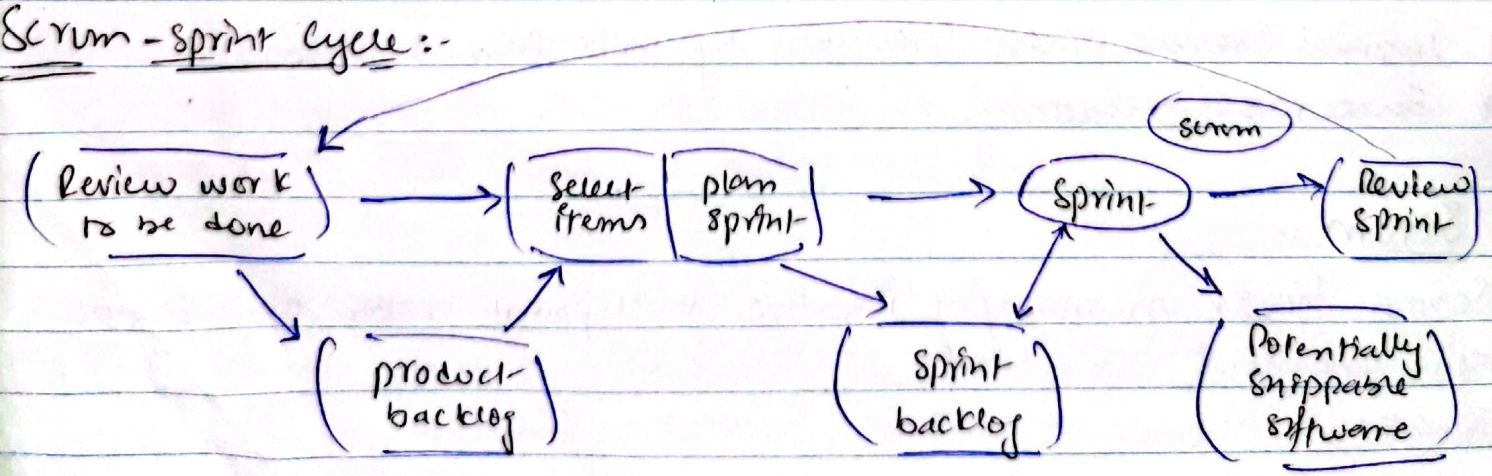
* Scrum focuses on managing iterative development rather than specific agile practices.
* phases in Scrum
  - ↳ Outlining phase → (general objectives and designing software architecture)
  - ↳ followed by series of sprint cycle → (each cycle develops an increment of system)
  - ↳ Project closure → (

Foundation for Advancement of Science & Technology

Scrum terminologies :-

- Development team → No more than 7 people for developing software
- Product backlog → This is a list of 'to-do' items which the scrum team must tackle → software → requirements, user stories or supplementary tasks such as architecture definition or user documentation.
- Product Owner → whose job is to identify product features or requirements, prioritize these for development and continously review the product backlog. Product owner can be customer / product manager / other stakeholder representative
- Scrum → daily meeting of scrum team → reviews progress and prioritize work to be done that day. → short f2f meetings.
- ScrumMaster → Ensures scrum process is followed and guides the team. Responsible for interfacing with rest of company.
- Sprint → development iteration. (2-4) weeks long.
- Velocity → Estimate of how much product backlog effort that a team can cover in a single sprint. → Understanding team's velocity provides a basis for measuring improving performance

Scrum - sprint cycle :-



The input for planning is product backlog

Foundation for Advancement of Science & Technology

Advantages

→ Freedom & Adaption
→ High-quality, low-risk product
→ Reduces development time upto 40%
→ Scrum customer satisfaction is very important
→ Reviewing the current sprint before moving to new one.
→ Manageable and understandable chunks.
→ Complete visibility → Team communication
→ Trust b/w customer and developers.

Disadvantages

↳ More efficient for small team size.
↳ No changes in the sprint.

* **Architectural Design** :- is a process for identifying the sub-systems making up a system and the framework for sub-system control and communication.
- Output of this design → software architecture
- Early stage of system design process.
- Link between specification and design processes (often carried out in parallel).
- Involves identifying major system components and their communication.

Software architecture can be designed at two level of abstraction
- Architecture in small is concerned with the architecture of individual programs. At this level, we are concerned with the way that an individual program is decomposed into components.
- Architecture in large is concerned with the architecture of complex enterprise systems that include other systems, programs, and program components. These enterprise systems are

distributed over different computers, which may be owned and managed by different companies.

- Advantages
- Stakeholder communication - Architecture may be used as a focus of discussion by system stakeholders.
- System analysis - Well-documented architecture enables the analysis of whether the system can meet its non-functional requirements.
- Large scale reuse - The architecture may be reusable across a range of systems or entire line of products.

* Architectural representation (Box and line diagrams)
- Very abstract - they do not show the nature of components relationships nor the externally visible properties of the sub-systems.
- However, useful for communication with stakeholders and for project planning.

* Architecture Design Processing :-
- ...
- Analyze requirements
- Define business use case (functional / non-functional)
- Choose architecture pattern / style.
- Communicate architecture to stakeholders (prototypes)
- Evaluate architecture
- Design detailed implementation architecture
- ...

Foundation for Advancement of Science & Technology

* Uses of Architectural Models :-
-) facilitating discussion about the system design.
-) Documenting an architecture that has been designed.

* Agility and Architecture Design
-) early stage of agile processes is to design an overall systems architecture
-) Architecture design activities takes place throughout the project.

* Architecture Reuse :-
-) Systems in the same domain often have similar architectures that reflect domain concepts.
-) Application product lines are built around a core architecture with variants that satisfy particular customer requirements.
-) Architecture of a system may be designed around one of more architectural patterns or styles.

* Architecture and System Characteristics :-
- Performance → Localise critical operations and minimise communications. Use large rather than fine-grain components.
- Security → Use a layered architecture with critical assets in the inner layers
- Safety → Localise safety-critical features in a small number of sub-systems.
- Availability → Include redundant components and mechanisms for fault tolerance.
- Maintainability → Use fine-grain, replaceable components.

Foundation for Advancement of Science & Technology

* Architectural views :-

→ Shows one view or perspective of the system.
→ how system is decomposed into modules, run-time procen interact, ways components are distributed across a network.

1) 4+1 model :-

→ A logical view, which shows the key abstraction in the system as object or object classes.
→ A process view, which shows how, at run-time, the system is composed of interacting processes.
→ A development view, which shows how the software is decomposed for development.
→ A physical view, which shows the system hardware and how software components are distributed across the processors in the system.
→ Related using use cases or scenarios (+1)

*) Architectural Patterns :-

→ mean pattern means of representing, sharing and reusing knowledge
→ Architectural pattern is a stylized description of a good design practice
→ pattern should include when they are useful or not.
→ pattern may be represented using tabular and graphical descriptions

* Model-View-Controller :-

→ Serves as a basis of interaction management in many web-based systems.
→ components :-

Foundation for Advancement of Science & Technology

→ The model is the central component of the pattern that directly manage the data, logic and rules of the application. Independent of user-interface.
→ A view can be any output representation of information, such as chart or diagram. Multiple views of same info are possible.
→ A controller accepts input and converts it to commands for the model of view.

•) Problem :-
→ display presented frequently changes in response to input or computation
→ Different needs how to view the program's information
→ need reflect data changes to all users in a way they want information
→ While making it easy to make changes to UI.

•) Solutions :-
→ Seperating the data and displaying, using three components → model → view → controller.

→ Advantages :-
•) views and controllers can be easily added, removed or changed
•) views can be added or changed during execution.
•) UI components can be changed, even at run-time.

→ Disadvantages :-
•) Views and controllers are often hard to seperate
•) Frequent updates may slow data display and degrade UI performance
•) UI components (view, controller) highly dependent on model components.

Foundation for Advancement of Science & Technology

* Layered Architecture :-

- Used for interfacing of sub-systems
- organizes the system into a set of layers each provide set of services.
- Supports the incremental development of sub-systems in different layers. When a layer interface changes, only the adjacent layer is affected.
- However, often artificial to structure systems in this way.

- **When Used?** → building new facilities on top of existing systems.
→ Development is spread across several teams with responsibility of a layer.
→ Requirement for multi-level security.

- **Advantages:-**
→ replacement of entire layers as long as interface is maintained.
→ Redundant facilities (eg authentication) can be provided to increase the dependability of the system.

- **Disadvantages:-**
→ high level layer may have to interact with lower level layer rather than layer immediately below it.
→ Performance can be a problem as it is processed at each layer because of multiple levels of interpretation of a service request.

* Repository Architecture :-

All data in a system is managed in a central repository that is accessible to all system components. Components do not interact directly, only through the repository.

Foundation for Advancement of Science & Technology

• When Used ?

→ System where large volumes of information are generated that has to be stored for a long time.
→ Data-Driven systems where inclusion of data in the repository triggers an action or tool.

• Advantages :-

→ Independent components.
→ Changes in one component can be propagated to all components.
→ Data can be managed consistently (backups) as it is all in one place.

• Disadvantages :-

→ Repository is a single point of failure
→ Inefficiency in organizing all communications through the repository.
→ Distributing repository across several computers may be difficult.

* Client-server Architecture :-

→ System is organized into services, with each service delivered from a seperate server. Clients are users of these services and access servers.

• When Used ?

→ Data in a shared database has to be accessed from a range of locations. Because servers can be replicated, may also used when the load on a system is variable.

• Advantages :-

→ Can be distributed across a network.

→ General functionality (e.g. printing service) can be available to all clients and doesn't need to be implemented by all services.

•) Disadvantages:-

→ Susceptible to DoS attack or server failure.

→ Unpredictable performance as it depends on the network as well as the system.

→ Management problems if owned by different organizations.