Alternate SDLC Models

Rapid Application Development

Features of RAD model

- It is based on prototyping and iterative development with no specific planning involved.
- Using the RAD model, software product is developed in a short period of time(60-90 days).
- The critical feature of this model is the use of powerful development tools and techniques.

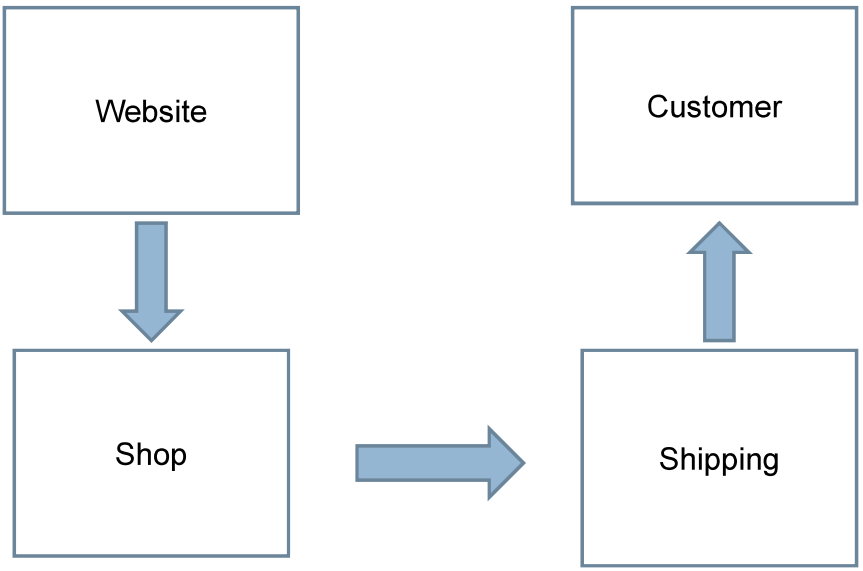
Features of RAD model

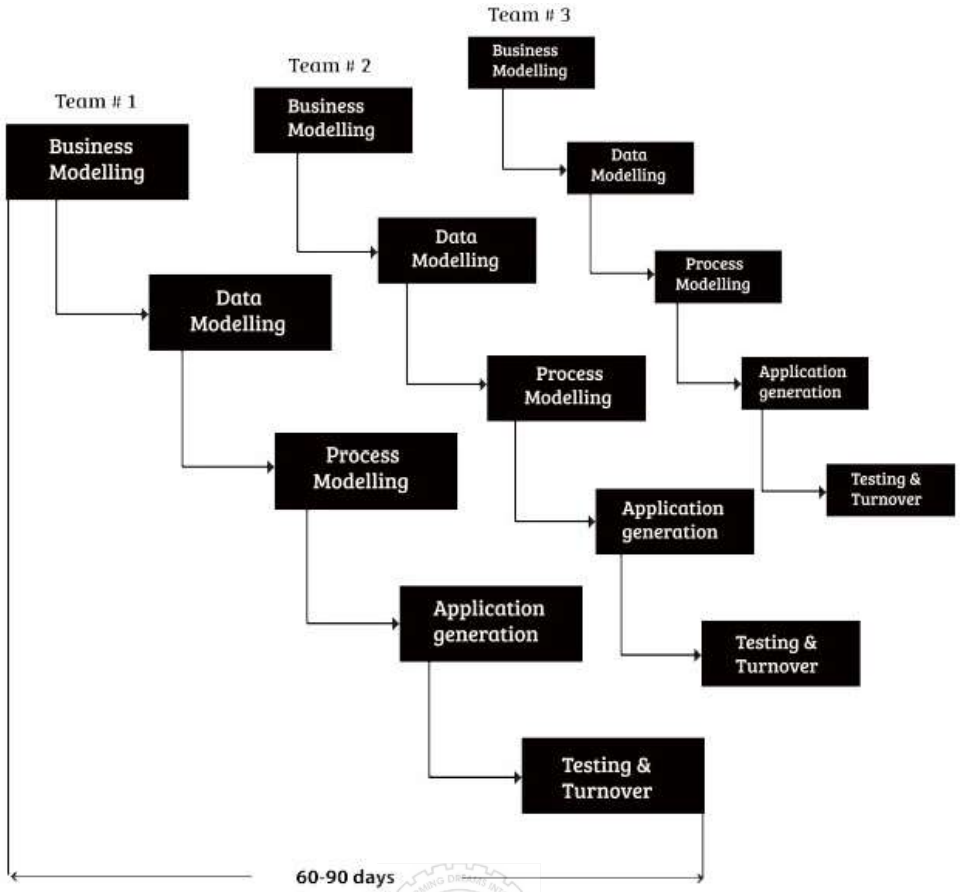
- The components or functions are developed in parallel as if they were mini projects.
- Gives the customer something to see and use and to provide feedback regarding the delivery and their requirements.
- Customer requirements through workshops or focus groups, early testing of the prototypes by the customer using iterative concept, reuse of the existing prototypes (components), continuous integration and rapid delivery.

Phases of RAD model

- Business Modeling
- Data Modeling
- Process Modeling
- Application Generation
- Testing & Turnover

Bhogilal's Online MedShop





Advantages of RAD model

- Reduced development time.
- Increases re-usability of components.
- Quick initial reviews occur.
- Encourages customer feedback.
- Integration from very beginning solves a lot of integration issues.

Disadvantages of RAD model

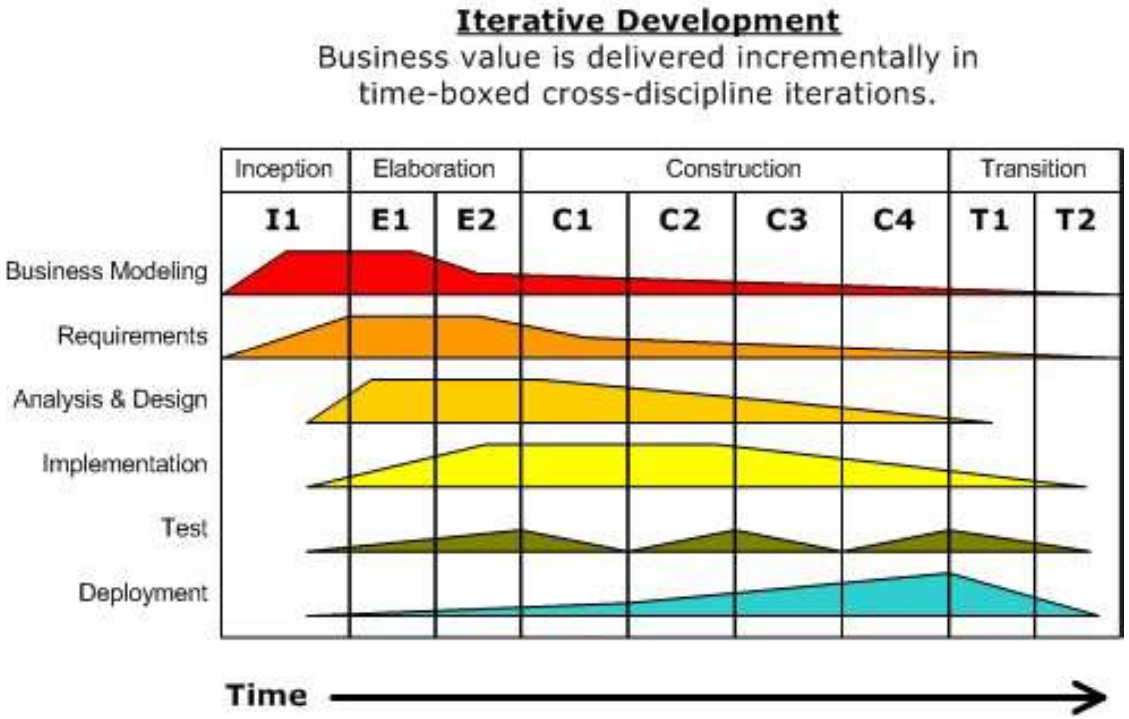
- Depends on strong team and individual performances for identifying business requirements.
- Only system that can be modularized can be built using RAD.
- Requires highly skilled developers/ designers.
- High dependency on modeling skills.
- For smaller projects, we cannot use the RAD model.
- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

Alternate SDLC Models

Rational Unified Process

The Rational Unified Process

- Hybrid and iterative software development framework.
- SDLC is Organized into Business phases(inception, elaboration, construction, and transition) &
- Technical Activities (requirements, analysis, and design, etc.)
- RUP incrementally grows an effective solution over multiple iterations



RUP Model Representation

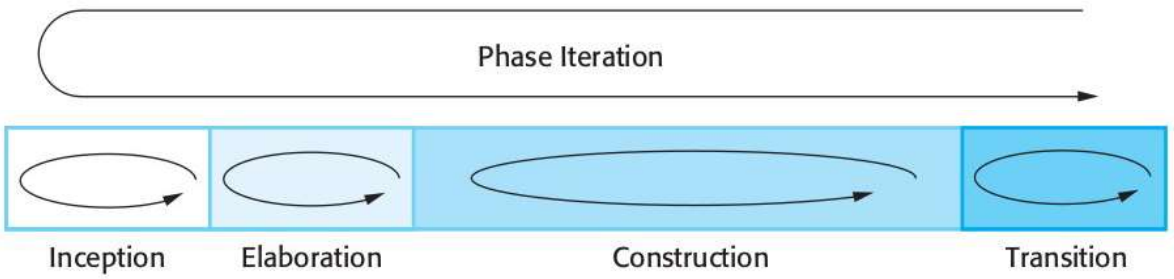
3 perspectives of RUP Model

- **A dynamic perspective that shows phases over time**
- **A static perspective that shows process activities**
- **A practice perspective that suggests good practice.**

RUP Phases

- **Inception** - Establish the business case for the system.
- **Elaboration** - Develop an understanding of the problem domain and the system architecture.
- **Construction** - System design, programming and testing.
- **Transition** - Deploy the system in its operating environment.

Phases in RUP



- **In-phase iteration** - Each phase is iterative with results developed incrementally.
- **Cross-phase iteration** - As shown by the loop in the RUP model, the whole set of phases may be enacted incrementally.

Static workflows in RUP

Workflow	Description
Business modelling	The business processes are modelled using business use cases.
Requirements	Actors who interact with the system are identified and use cases are developed to model the system requirements.
Analysis and design	A design model is created and documented using architectural models, component models, object models and sequence models.
Implementation	The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.

Static workflows in RUP

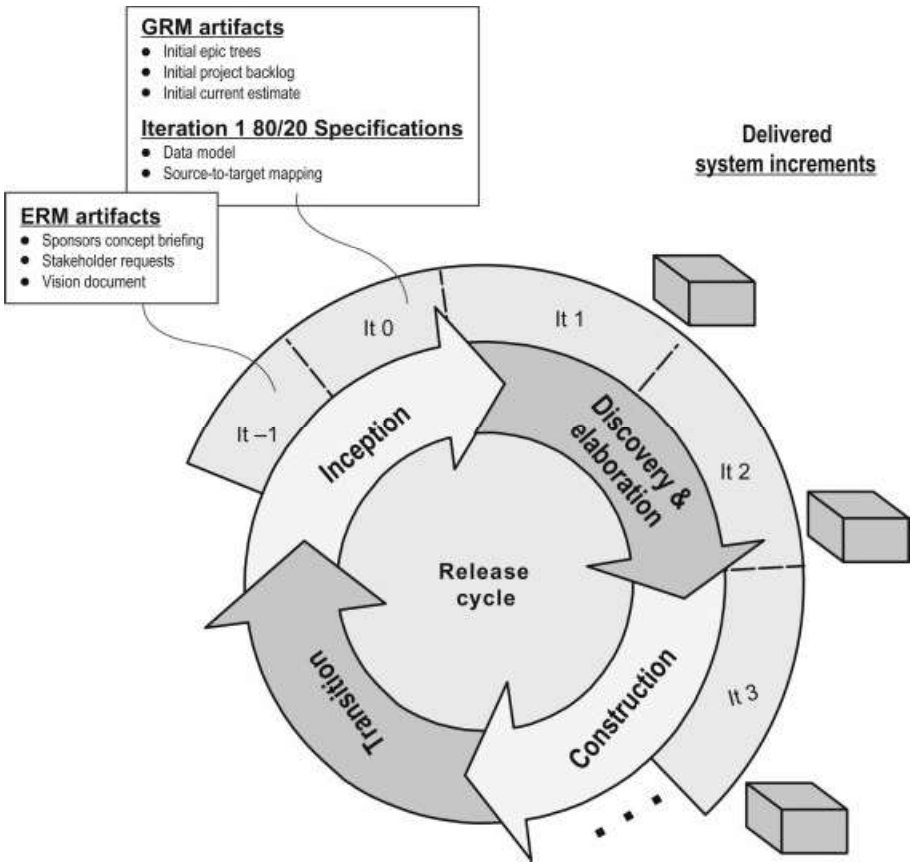
Workflow	Description
Testing	Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
Deployment	A product release is created, distributed to users and installed in their workplace.
Configuration and change management	This supporting workflow managed changes to the system.
Project management	This supporting workflow manages the system development.
Environment	This workflow is concerned with making appropriate software tools available to the software development team.

RUP Good Practices [1/2]

- **Develop software iteratively**: Plan increments of the system based on customer priorities and develop the highest-priority system features first.
- **Manage requirements Explicitly**: Document the customer's requirements and keep track of changes to these requirements.
- **Use component-based architectures**: Structure the system architecture into components.

RUP Good Practices [2/2]

- **Visually model software**: Use graphical UML models to present static and dynamic views of the software.
- **Verify software quality**: Ensure that the software meets the organizational quality standards.
- **Control changes to software**: Manage changes to the software using a change management system and configuration management procedures and tools.



Alternate SDLC Models

Agile Software Development

Agile Methods

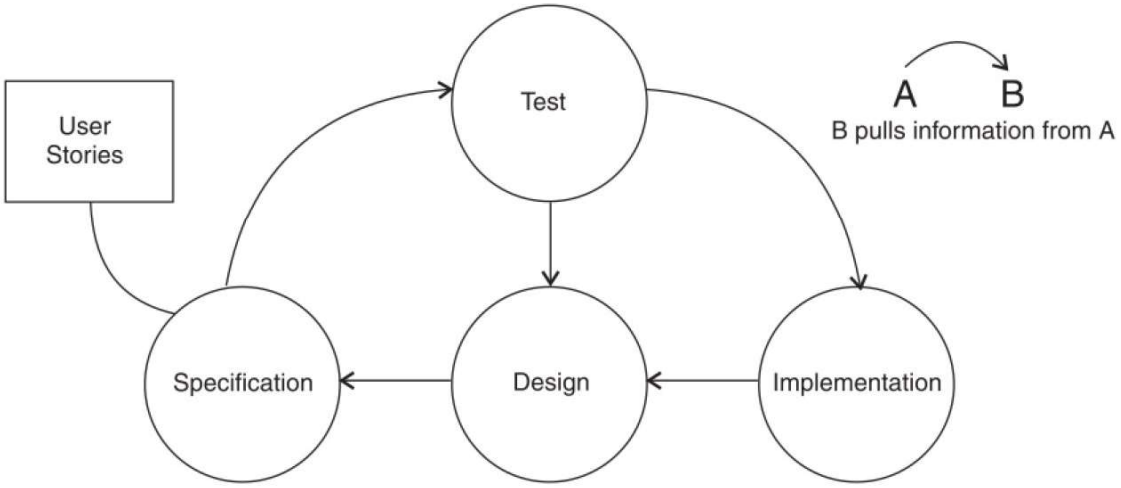
Agile methods:

- Focus on the **code rather than the design**
- Are based on an **iterative approach** to software development
- Are intended to **deliver working software quickly** and evolve this quickly to meet changing requirements.
- The aim of agile methods is to **reduce overheads** in the software process (e.g. by limiting documentation) and to be able to **respond quickly to changing requirements** without excessive rework.

Agile Manifesto

- We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
- *Individuals and interactions* over *processes and tools*
- *Working software* over *comprehensive documentation*
- *Customer collaboration* over *contract negotiation*
- *Responding to change over following a plan*
- That is, while there is value in the items on the right, we value the items on the left more.

Agile Development Process



The principles of Agile methods

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

Agile method applicability

- Product development where a software company is developing a **small or medium-sized product** for sale.
- Custom system development within an organization, where there is a **clear commitment from the customer to become involved in the development process** and where there are not a lot of external rules and regulations that affect the software.
- Because of their focus on small, tightly-integrated teams, there are **problems in scaling agile methods** to large systems.

Problems with agile methods

- It can be difficult to keep the **interest of customers** who are involved in the process.
- Team members may be unsuited to the **intense involvement** that characterizes agile methods.
- **Prioritizing changes** can be difficult where there are multiple stakeholders.
- **Maintaining simplicity** requires extra work.
- **Contracts may be a problem** as with other approaches to iterative development.

Alternate SDLC Models

Extreme Programming

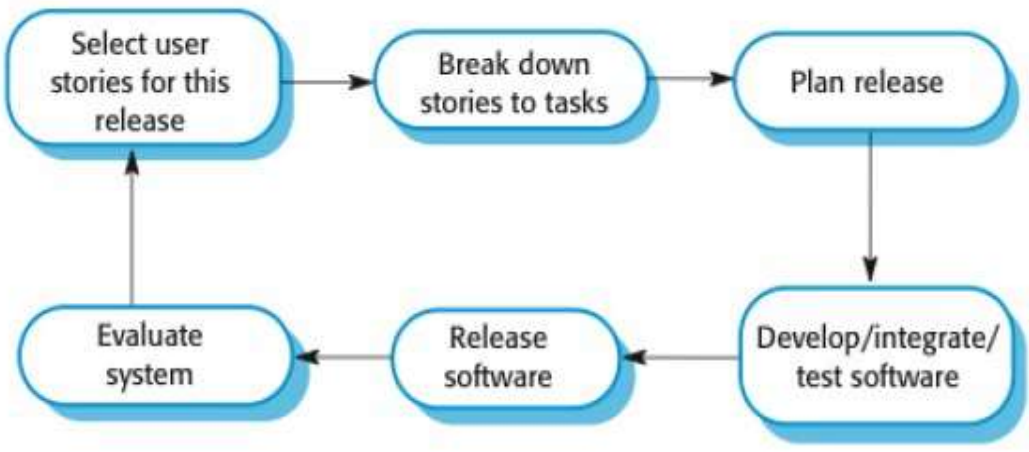
Extreme programming

- Best-known and most widely used agile method.
- Extreme Programming (XP) takes an 'extreme' approach to iterative development.
- Increments are delivered to customers every 2-4 weeks
- All tests must be run for every build and the build is only accepted if tests run successfully.

XP and Agile practices

- **Incremental development** is supported through small, frequent system releases.
- **Customer involvement** means full-time customer engagement with the team.
- **People not process** through pair programming, collective ownership and a process that avoids long working hours.
- **Change** supported through regular system releases.
- **Maintaining simplicity** through constant refactoring of code.

XP Release cycle



XP Practices

Principle or practice	Description
Incremental planning	Requirements are recorded on story cards and the stories to be included in a release are determined by the time available and their relative priority. The developers break these stories into development 'Tasks'. See Figures 3.5 and 3.6.
Small releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple design	Enough design is carried out to meet the current requirements and no more.
Test-first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code continuously as soon as possible code improvements are found. This keeps the code simple and maintainable.

XP Practices

Pair programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.
Continuous integration	As soon as the work on a task is complete, it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of overtime are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
On-site customer	A representative of the end-user of the system (the customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

Alternate SDLC Models

SCRUM

Scrum

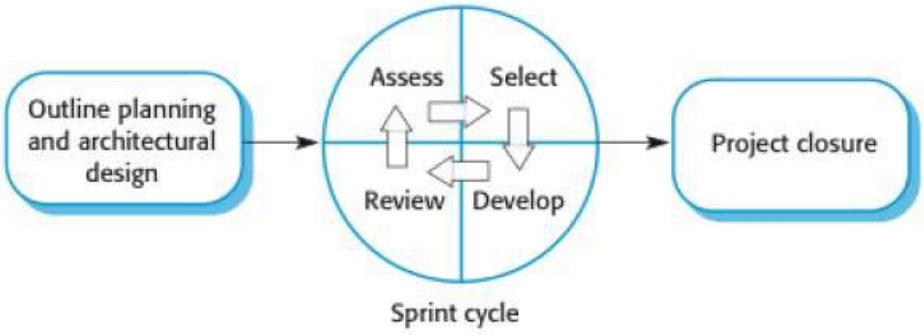
- Scrum focus is on managing iterative development rather than specific agile practices.
- Provides a project management framework.
- It is centered around a set of sprints, which are fixed time periods when a system increment is developed.

Scrum Phases

There are three phases in Scrum.

- The initial phase is an outline planning phase where you establish the general objectives for the project and design the software architecture.
- This is followed by a series of sprint cycles, where each cycle develops an increment of the system.
- The project closure phase wraps up the project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project.

SCRUM Process



Sprint cycle

- Sprints are fixed length, normally 2–4 weeks.
- The starting point for planning is the product backlog, which is the list of work to be done on the project.
- The selection phase involves all of the project team who work with the customer to select the features and functionality to be developed during the sprint.

Sprint cycle

- Once these are agreed, the team organize themselves to develop the software.
- Team is isolated from the customer and the organization, with all communications channeled through the so-called 'Scrum master'.
- Scrum master protects the development team from external distractions.
- At the end of the sprint, the work done is reviewed and presented to stakeholders.
- The next sprint cycle then begins.

Teamwork in SCRUM

- The 'Scrum master' is a facilitator who arranges daily meetings, tracks the backlog of work to be done, records decisions, measures progress against the backlog and communicates with customers and management outside of the team.
- The whole team attends short daily meetings where all team members share information, describe their progress since the last meeting, problems that have arisen and what is planned for the following day.

Scrum benefits

- The product is broken down into a set of manageable and understandable chunks.
- Unstable requirements do not hold up progress.
- The whole team have visibility of everything and consequently team communication is improved.
- Customers see on-time delivery of increments and gain feedback on how the product works.
- Trust between customers and developers is established and a positive culture is created in which everyone expects the project to succeed.