1. **System (Software) development methodology**

**Definition:**

It is a framework that used to structure, plan, and control the process of developing an information system in software engineering.

**How does it help**

It contains detailed steps to be carried out at specific situations / specific type of project

**Importance**

* To ensure consistency in management of projects
* To ensure the best techniques applied to carry out a project.
* Reduce time & cost
* To improve the quality of work & system
* Meet user requirement

1. **System development planning**

**Tools used for Resource Planning**

* PERT Chart & Gantt Chart – For project activities and human resources planning
* Work Break-down Structure - For tasks and multiple resources assigned to each.
* Hierarchy Task Analysis - For task restructuring

**Outsourcing \*\*\*\*\***

|  |  |
| --- | --- |
| Advantages | Explain |
| Get More Experts | By outsourcing particular tasks, companies are often able to substantially improve performance by drawing on the niche skills of experts in certain fields. |
| Things Get Done Fast | You can get things done quicker by-passing time-consuming tasks by outsourcing. |
| You’re Able to Focus on What Matters | By passing on supporting processes, you’ll be able to concentrate your skills on strengthening and improving the core processes that help make your business tick. |
| You Can Share Some Risk | By outsourcing certain campaigns or processes on to experts in their respective fields, you will benefit from their enhanced ability to plan and mitigate potential risks. |
| You Can Reduce Costs | It is cheaper to outsource specific tasks than to hire permanent employees. |
| You Can Simplify Project Management | Outsourcing companies usually provide their customers with dynamics. Intuitive platform. Through the platform, you can effectively manage the work being performed. |
| Disadvantage | Explain |
| Lose Some Control | You’re losing control of how those tasks are being monitored and performed. |
| Security Risks | Because outsourcing requires data sharing, the privacy or corporate security of others may be threatened. |
| Reduce Quality Control | Outsourcing can be stimulated by time and profits to lower quality standards. |

1. **Risk management (Examples of risk)**

**Definition**

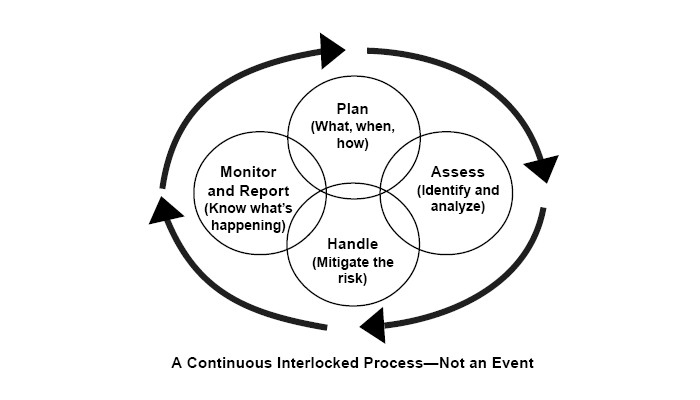
Process of identifying potential risk to the project / products and getting prepared for it.

**Why need?**

Many projects failed due to

* Developer did not identify all risks
* Developer Ignoring risks
* Developer did not prepare ‘backup plan’ for risk
* Developer did not monitor risk.

**Process or risk management**



**Risk management Strategy suggests**

Risk Transfers

* Accept that the risk **MAY** happen. Transferring the risk to a vendor or customer (who are aware of the risk but willing to face it if it comes).

Risk Avoidance

* Accept that the risk **MAY** happen and taking alternative path to avoid the risk from happening.
* Original design may change.

Risk Reduction

* Accept that the risk **WILL** happen and taking additional steps to reduce the risk form occurring.
* May increase cost and delay in delivery time.

Risk Acceptance

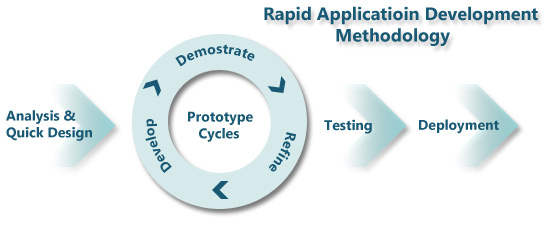
* Accept that the risk **WILL** happen and implement total solution.
* Costly and time consuming.

1. **process-oriented management \*\*\*\*\***

**Rapid Application Development (RAD)** - used for small and fast projects. Within days/weeks.

**Implement**

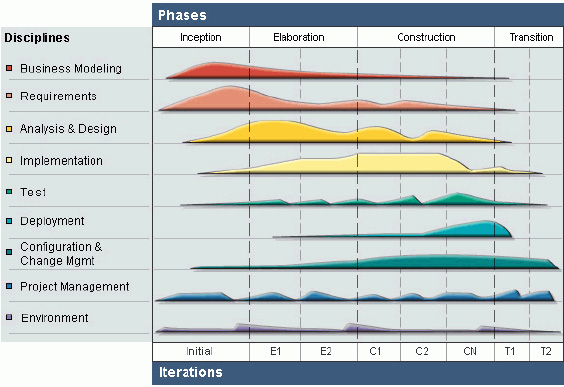
**Apply**:



* Expert developers used
* Uses tools (CASE Tools) for faster development and testing
* Uses minimal planning, analysis and documentation
* Uses Prototype for user feedback & review, product development
* Iterative and Incremental design approach with prototyping
* Users are involved in development

**Rational Unified Process (RUP)** - emphasis on quality of product and process. “plan a little, design a little, and code a little”

**Implement**



**Apply**

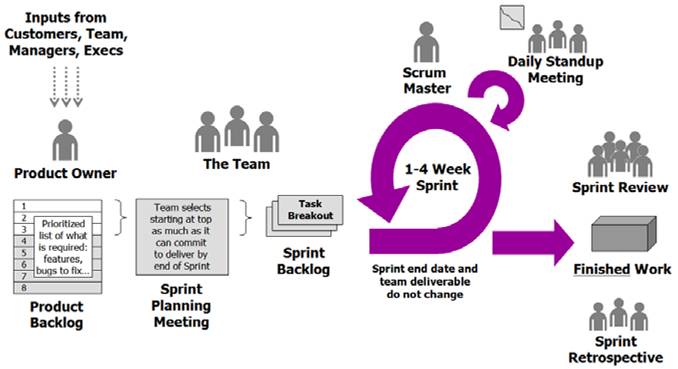
Extensive and exclusive use of UML, direct support for OO Programming.

**Takes a holistic approach of the system;**

* Architecture of the system determined
* Task broken-down to smaller components.
* Iterative & incremental approach applied to do all tasks.

**SCRUM -** A ‘team-work’ based methodology.

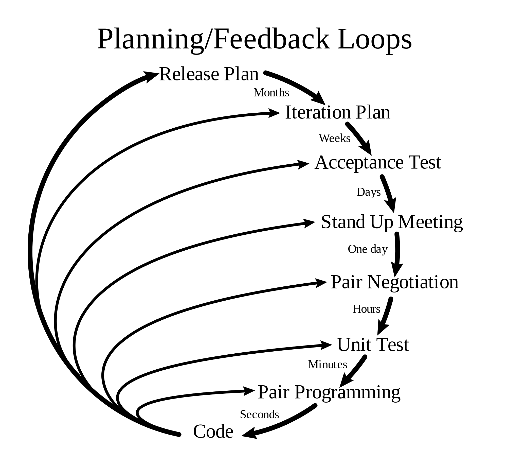
**Implement**

****

**Apply**

* **User Stories** – developer & user come up with general concept of the system
* **Product Owner** - determines that the business requirements are met
* **Development Teams** are from mixed skills
* **Scrum Master** helps to solve the ‘gaps’ in the project such as solving problems or giving ideas.
* **Sprint** – a cycle of a task to be performed in the project.
* **Backlogs** – this need to be for the project to be completed.

**Extreme Programming (X**P**)** - For advance / heavy coding projects

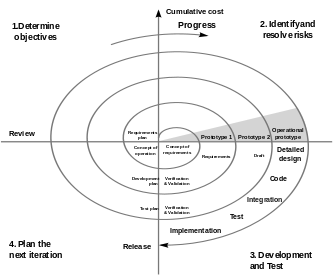


**Apply**

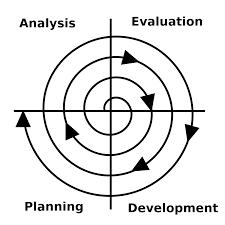
* Small and frequent "releases" in short development cycles
* Development team – fully integrated. Pair programming, stand-up meetings
* Test driven development. Close user involvement, testing
* Accept changing requirement at any time
* Simplicity in everything. Tries to simplify all processes
* Produce high quality software
* And other agile principle.

**Spiral Methods** - used for project which has many sections but need to be linked later.

**Implement**



**Apply**

* **Planning Phase**
  + Requirements are gathered.
  + Specification are set.
* **Risk Analysis**
  + Identify risk and alternate solutions.
  + A prototype is produced
* **Engineering Phase**
  + Software is developed
  + Through Testing carried out.
* **Evaluation phase**
  + Allows the customer to evaluate the output of the project to date before the project continues to the next spiral.

1. **Comparing\*\*\*\*\*\***

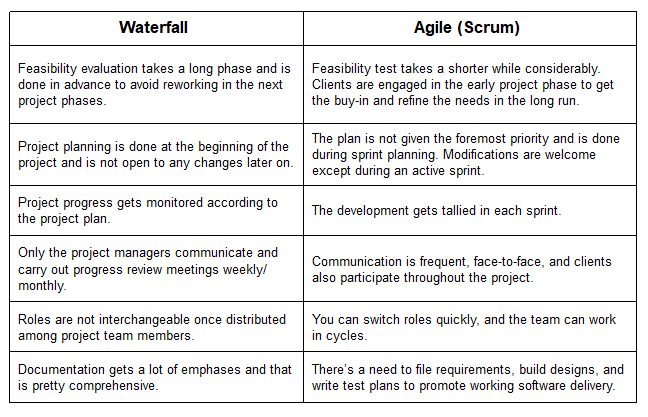
(choose right methodology depends on case? why suggest? how help you to choose suitable methodology compare agile structure and characteristic, modern)

**Selection depends on:**

* The type of problems and suggested solution
* The type of project
* Speed of the project
* The type of products
* Requirements are fixed or can be often changing.
* Size and budget of project
* Knowledge of developer
* Support for a particular methodology is easily obtained
* Availability of users throughout the project.



**Structure and characteristic of agile and traditional methodology**



1. **Agile\*\*\*\*\*\***

Agile Methods are a set of Moderns IS Methodologies which shares some of the Agile Principles.

**Principles:**

* **Customer Satisfaction**
  + By fast and frequent delivery of products
  + Welcome changing requirements, even late in development
* **Teamwork**
  + By Face-to-face communication with all people involved
  + Motivate and trust developers
* **Fast Development**
  + Break bigger system into small and manageable components
  + Close monitoring of development
* **High Product Quality**
  + Maintain good design and simplicity
  + Adopt to latest technologies.

**Advantages**

* Customer satisfaction with frequent delivery of working product.
* Gives customers/users ‘power’ to change their mind anytime and send new requirements.
* Gives more ‘control’ to core developers to make decisions
* Emphasize the use of latest design and technologies
* Encourage close communication and teamwork.

**Disadvantages**

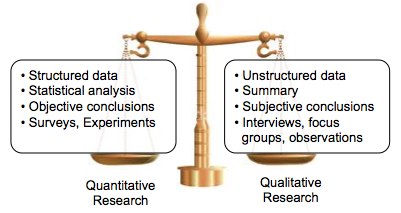
* Users/customers not available at all time.
* Developer - difficult to determine final cost and development time as requirement keep changing
* Developer – difficult to plan and deliver workable product frequently.
* Experts developers and CASE Tools are expensive
* Often lack comprehensive documentation

1. **Data analysis**

The process of data analysis

1. Collecting and storing data
2. Cleaning data
3. Analyzing data
4. Creating output information
5. Conclusion of analysis

**Quantitative vs Qualitative research / analysis**



1. **System design**

**Why need?**

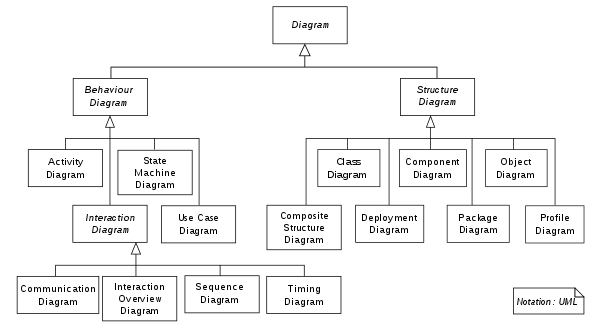
Reduce Your Design Debt; Eliminate Inconsistencies; Speed Up Your Process; Make Your Customers’ Lives Easier

**Purpose**

* To creating a solution to a problem in hand
* To breakdown and understand complex concepts and process.
* To create a product safe and secure for use (HCI)
* To add vital details to product
* To come-up with efficient and effective product / solution
* To ensure product comply with set specifications and standards.
* To implement modern technology into product
* To test product early and avoid costly rework

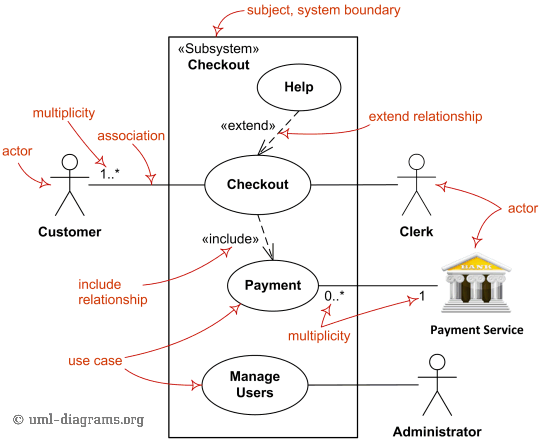
1. **UML modeling**

UML is the modeling techniques to visualize the design of a system.



UML can be used to design, discuss, present and event test a software component before its built.

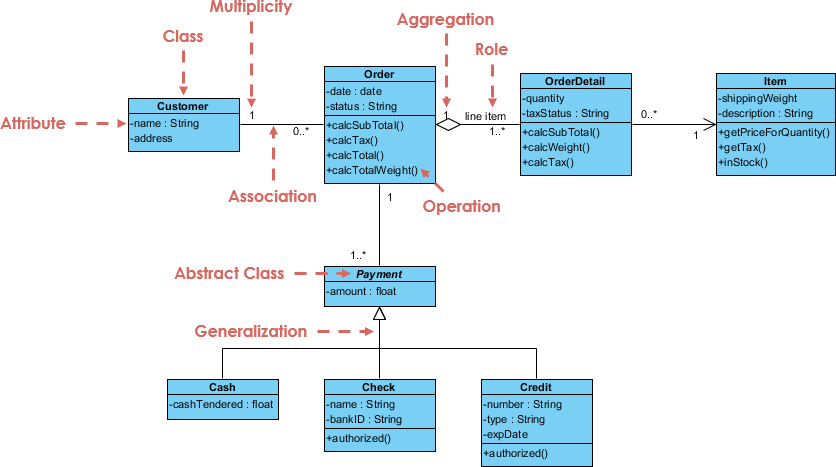
* **Use case diagram \*\*\*\*\***



Use cases focus on the users of the system, not the system itself, thus the real system needs are brought to light early on.

**Advantages**:

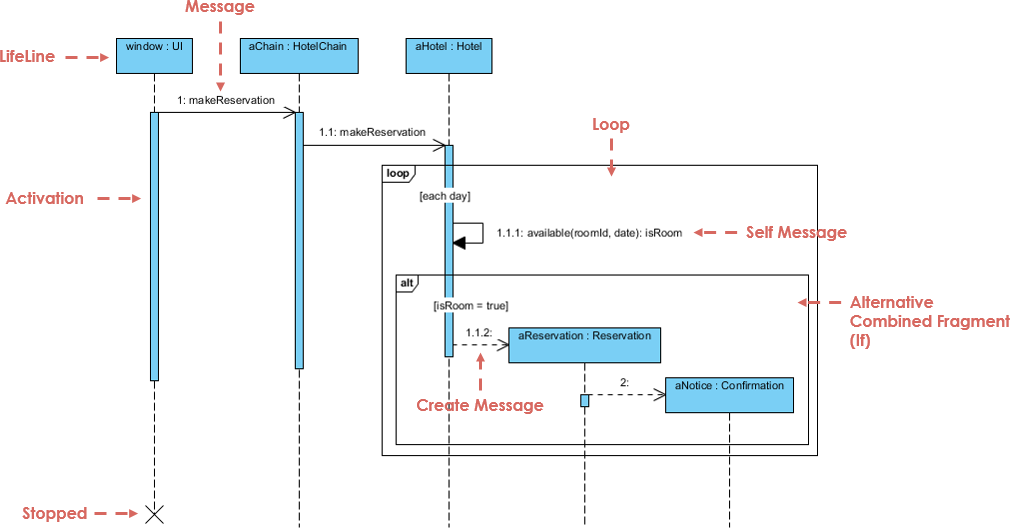
* The feedback can be obtained at a very early stage of the development from the customers and the end users.
* It requires the identification of exceptional scenarios for the use cases. This helps in discovering subtle alternate requirements in the system.
* The use case model can be utilized in several other aspect of software development as well.
* The use case diagram provides a comprehensive summary of the whole software system in a single illustration.
* **Class diagram \*\*\*\*\***



Class diagrams describe systems by illustrating attributes, operations and relationships between classes. Unified Modeling Language (UML) calls them structure diagrams. They work according to the principles of object orientation. This orientation describes how objects interact with each other.

**Advantages**:

* Class diagrams give you a sense of orientation. They provide detailed insight into the structure of your systems. At the same time, they offer a quick overview of the synergy happening among the different system elements as well as their properties and relationships.
* Class diagrams are simple and fast to read. With the right software they are also easy to create. They are the foundation for creating systems.
* **Sequence diagram**



Model high-level interaction between active objects in a system

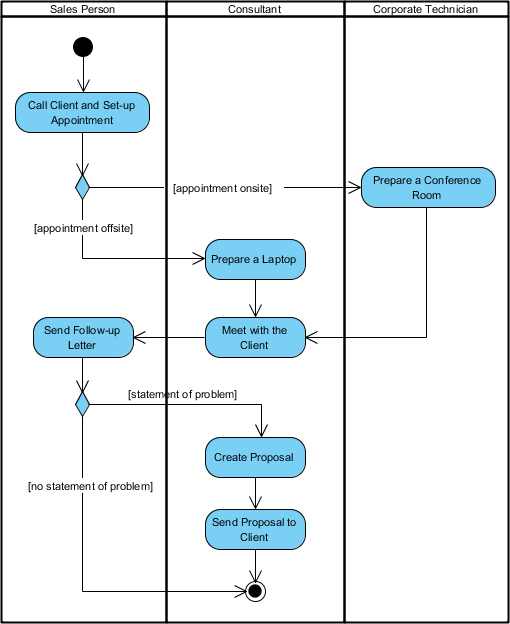
Model the interaction between object instances within a collaboration that realizes a use case

Model the interaction between objects within a collaboration that realizes an operation

Either model generic interactions (showing all possible paths through the interaction) or specific instances of a interaction (showing just one path through the interaction)

**Advantages**:

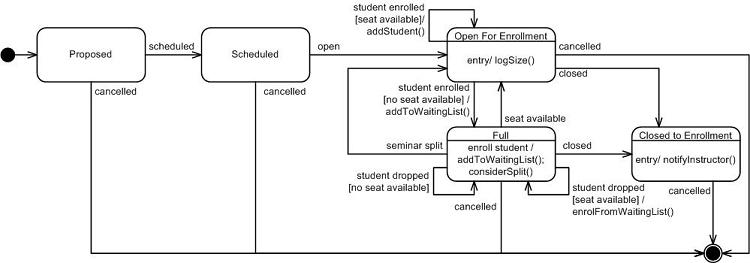
* Help you discover architectural, interface and logic problems early.  they allow you to flesh out details before having to implement anything, sequence diagrams are useful tools to find architectural, interface and logic problems early on in the design process.
* Collaboration tool. Sequence diagrams are valuable collaboration tools during design meetings because they allow you to discuss the design in concrete terms.
* Documentation. sequence diagrams can be used to document the dynamic view of the systemdesign at various levels of abstraction, which is often difficult to extract from static diagrams or even the complete source code.
* **Activity diagram**



Activity diagrams describe the actual work flow behavior of a system in Information Technology.

**Advantages**:

* UML modeling language included that these diagrams are normally easily comprehensible for both analysts and stakeholders.
* In UML for the IT Business Analyst, “The activity diagram is the one most useful to the IT BA for depicting work flow [because] it is simple to understand-both for BAs and end-users.”
* Since they are among the most user-friendly diagrams available, they are generally regarded as an essential tool in an analyst’s repertoire.
* Additionally, as stated above, activity diagrams allow an analyst to display multiple conditions and actors within a work flow through the use of swimlanes. Swimlanes, however, are optional as a single condition or actor is normally displayed without them.
* **State machine diagram (focus on object)**



The state design pattern is a scheme used to model changes in the state (or states) of an object.

**Advantages**

* Its capability to minimize conditional complexity.
* It eliminates the need for it and switches statements on objects with different behavioral requirements, unique to different state transitions.
* Representing an object’s state using a finite state machine diagram simplifies the conversion of the diagram into a state design model’s types & method.

1. **Implementation**

**Software testing Objective?**

* Finding defects / bugs.
* Prevent defects (avoid expensive recovery)
* Improving level of quality (code and product)
* To make sure that the end result meets the business and user requirements (in specifications).

**Approach**

* **Static Testing**
  + Test and find defects without executing code.
  + Done during verification process (verify requirements)
  + Includes techniques - reviewing of the documents, static analysis, reviewing, walkthrough, inspection, etc.
* **Dynamic Testing**
  + Software code is executed.
  + Done during validation process (satisfy customers)
  + Includes test types - Unit Testing, Integration Testing, System Testing, etc.
  + Recommended – automated testing tools.

**Black testing vs White testing**

* White Box Testing
  + Testing where all INPUT, PROCESS (code functions) and OUTPUT is seen.
  + Usually test by programmer.
* Black Box testing
  + Testing where only INPUT and OUTPUT is seen, PROCESS (internal code and functions) are hidden.
  + Usually test by User / Customer.
* Stub Testing
  + Testing just one line of execution, normally to check presence of data

**Testing level**

* **Unit testing**
  + component testing
  + Tests that verify the functionality of a specific section of code, usually at the function level.
  + (In an object-oriented environment, this is usually at the class level)
  + One function might have multiple tests
  + Ensure that the building blocks of the software work independently from each other.
  + Variation of white-box testing
* **Integration Testing**
  + Any type of software testing that seeks to verify the interfaces between components against a software design.
  + Works to expose defects in the interfaces and interaction between integrated components (modules).
  + Larger software components test the architectural design
* **Component Interface Testing**
  + To check the handling of data passed between various units, or subsystem components
  + Help explain unexpected performance in the next unit.
  + Variation of black-box testing
* **System Testing**
  + Tests a completely integrated system to verify that it meets its requirements
  + Variation of black-box testing

1. **Deployment**

**Deployment plan**

1. Determining Deployment Options
   1. Where will the system / software will be delivered?
2. Packaging Software / Systems
   1. What and how the components will be delivered to the users?
3. Deployment Scheduling
   1. Inform stakeholders of deployment event, who and when?
4. System change-over strategies
   1. How the old system will be replaced with new?
5. Integrating System Components
   1. Installing equipment / connecting servers, etc
6. Post-Deployment (On-Site) Testing
7. Provide Training to users (if needed)

**Importance of plan**

Improve the quality of your software and ensure a better product

**Change-over strategy?**

1. Direct Changeover

Direct changeover, also referred to as immediate replacement, tends to be the least favorite of the changeover techniques. In a direct changeover, the entire system is replaced in an instant.

1. Parallel Changeover

In a parallel changeover, the new system runs simultaneously with the old for a given period of time.

1. Pilot Changeover

With a pilot changeover, the new system is tried out at a test site before launching it company-wide.

Old camps use old system, new camps use new system

1. Phased Changeover

In a phased changeover, the new system is implemented one stage at a time.

Implement one stage at a time.

