AIM

The aim of this project is to make our home a smart home, making life easier for us. With the upcoming technologies, our main vision is to make a house with all the automatic facilities inside it.

MISSION

Our mission is to make our daily lives easier for us.
Also, switching off our electrical appliances will also lead to conservation of energy
thereby also leading to a healthier environment.

Google Assistant

We are planning to let our mobiles control our house appliances so that even we forget to put off the lights and fans or other appliances then our Google Assistant can recognise and do so for us.

ARDIUNO

We will be doing a basic coding to convert our basic home Elements to a smart home appliances using a wifi module. Connecting then to a single network and then controlling the home by just the touch of our fingers or just by using google assistant or some other AI program

FUNCTIONAL UNITS

- Lights and Fans control
- Entertainment Control
- Lights colour and opacity control
- Intruder alert (Pending to plan)

INTRODUCTION TO SDLC

SDLC DEFINITION: Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

The life cycle defines a methodology for improving the quality of software and the overall development process.

- SDLC is the acronym of Software Development Life Cycle.
- It is also called as Software Development Process.
- SDLC is a framework defining tasks performed at each step in the software development process.

STAGES IN SDLC

Stage 1: Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

Stage 2: Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

Stage 3: Designing the Product Architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

Stage 4: Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

Stage 5: Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Stage 6: Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

SDLC MODELS

There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred as Software Development Process Models. Each process model follows a Series of steps unique to its type to ensure success in the process of software development.

Following are the most important and popular SDLC models followed in the industry −

- Waterfall Model
- Iterative Model
- Spiral Model
- V-Model
- Big Bang Model

Other related methodologies are Agile Model, RAD Model, Rapid Application Development and Prototyping Models.

WATERFALL MODEL

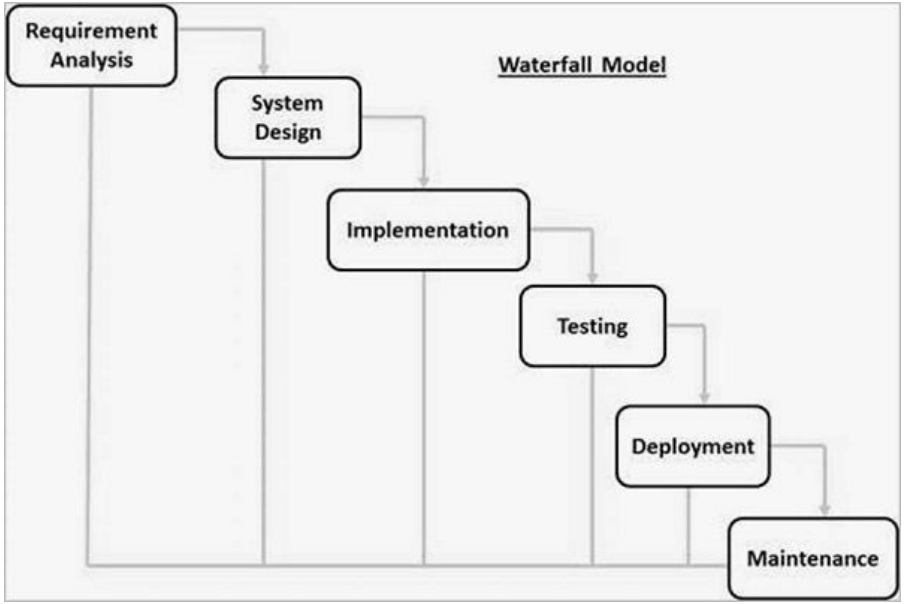
Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

APPLICATIONS :

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

PHASES :

- Requirement Gathering and analysis
- System Design
- Implementation
- Integration and Testing
- Deployment of system
- Maintenance



ADVANTAGES:

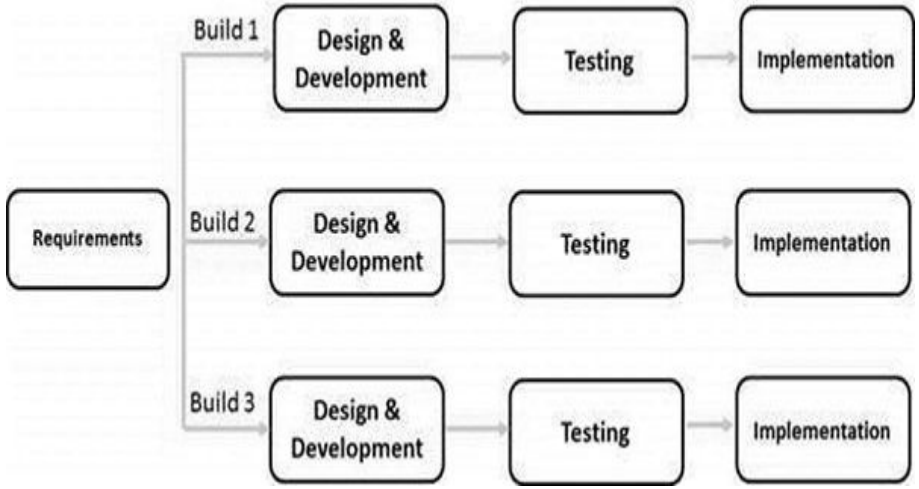
- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

DISADVANTAGES:

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.

ITERATIVE MODEL

Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).



APPLICATIONS :-

- Requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
- There is a time to the market constraint.
- A new technology is being used and is being learnt by the development team while working on the project.
- Resources with needed skill sets are not available and are planned to be used on contract basis for specific iterations.
- There are some high-risk features and goals which may change in the future.

ADVANTAGES:

- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Testing and debugging during smaller iteration is easy.
- Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.
- Easier to manage risk - High risk part is done first.

DISADVANTAGES:

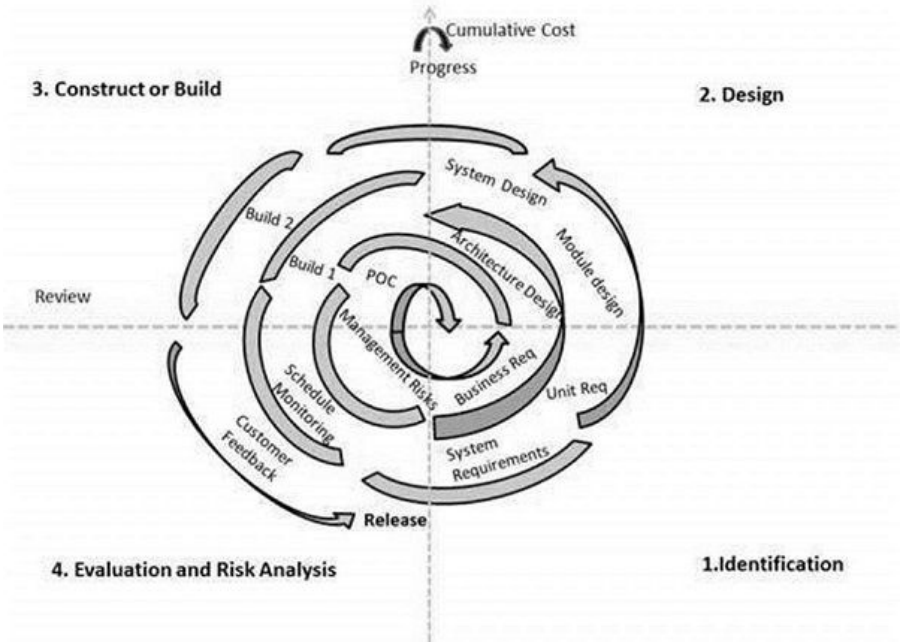
- More resources may be required.
- Although cost of change is lesser, but it is not very suitable for changing requirements.
- More management attention is required.
- Not suitable for smaller projects.
- Management complexity is more.
- End of project may not be known which is a risk.
- Highly skilled resources are required for risk analysis.

SPIRAL MODEL

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis. The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.

PHASES:-

- IDENTIFICATION
- DESIGN
- CONSTRUCT OR BUILD
- EVALUATION AND ANALYSIS



APPLICATIONS:-

- When there is a budget constraint and risk evaluation is important.
- For medium to high-risk projects.
- Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
- Customer is not sure of their requirements which is usually the case.
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phases to get enough customer feedback.
- Significant changes are expected in the product during the development cycle.

ADVANTAGES:-

- Changing requirements can be accommodated.
- Allows extensive use of prototypes.
- Requirements can be captured more accurately.
- Users see the system early.
- Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management.

DISADVANTAGES:-

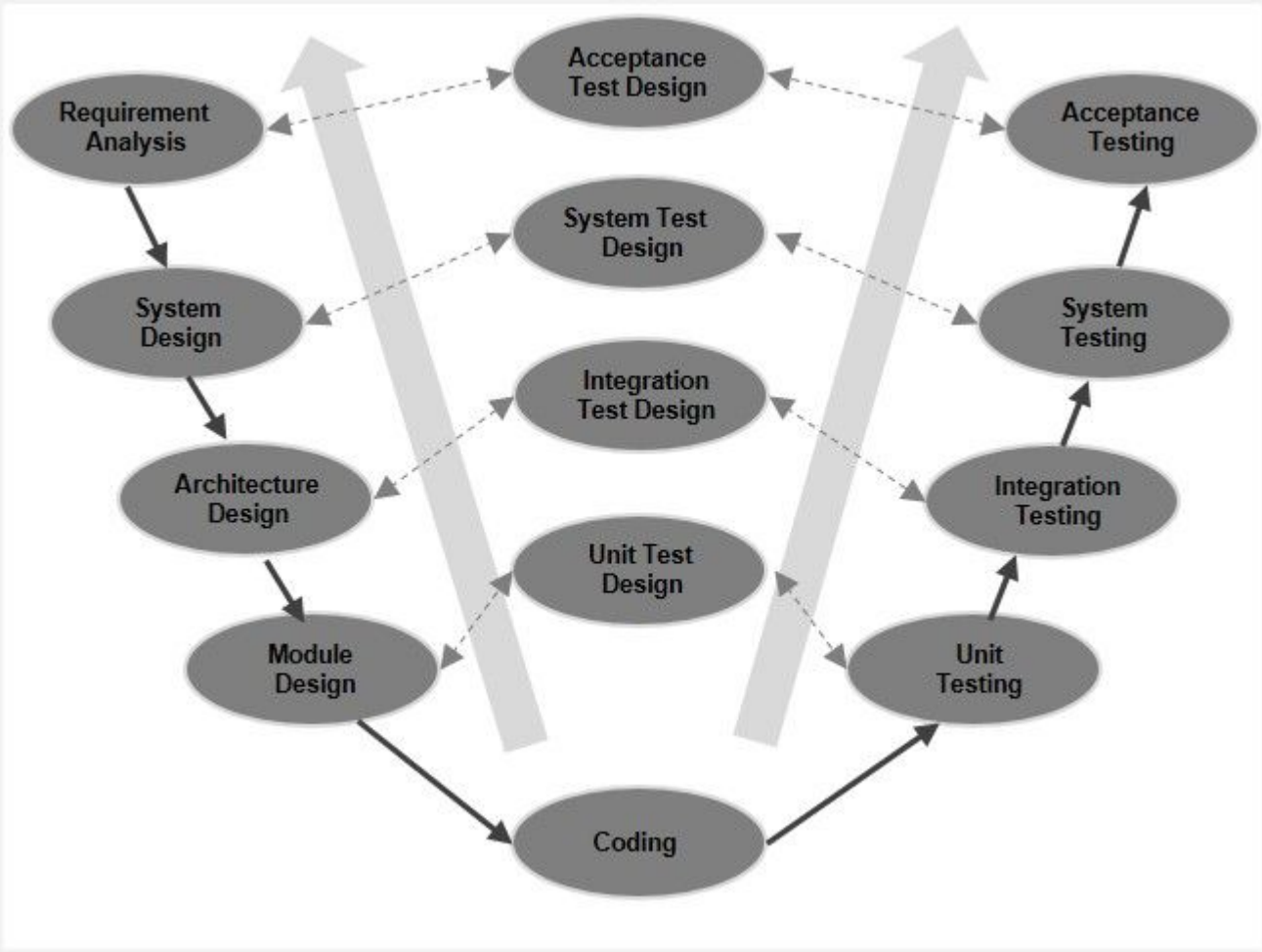
- Management is more complex.
- End of the project may not be known early.
- Not suitable for small or low risk projects and could be expensive for small projects.
- Process is complex
- Spiral may go on indefinitely.
- Large number of intermediate stages requires excessive documentation.

V-MODEL

The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. It is also known as Verification and Validation model.

The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase. This is a highly-disciplined model and the next phase starts only after completion of the previous phase.

Under the V-Model, the corresponding testing phase of the development phase is planned in parallel. So, there are Verification phases on one side of the ‘V’ and Validation phases on the other side. The Coding Phase joins the two sides of the V-Model.


Requirement Analysis
Acceptance Test Design
Acceptance Testing
System Test Design
System Design
System Testing
Integration Test Design
Architecture Design
Integration Testing
Unit Test Design
Module Design
Unit Testing
Coding

VERIFICATION PHASES

- Business requirement analysis
- System design
- Architectural design
- Module design
- Coding phase

VALIDATION PHASES

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing

APPLICATIONS

- 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.

ADVANTAGES:-

- This is a highly-disciplined model and Phases are completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Simple and easy to understand and use.
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.

DISADVANTAGES:-

- High risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.
- Once an application is in the testing stage, it is difficult to go back and change a functionality.
- No working software is produced until late during the life cycle.

PROPOSED MODEL FOR OUR PROJECT

AGILE MODEL :

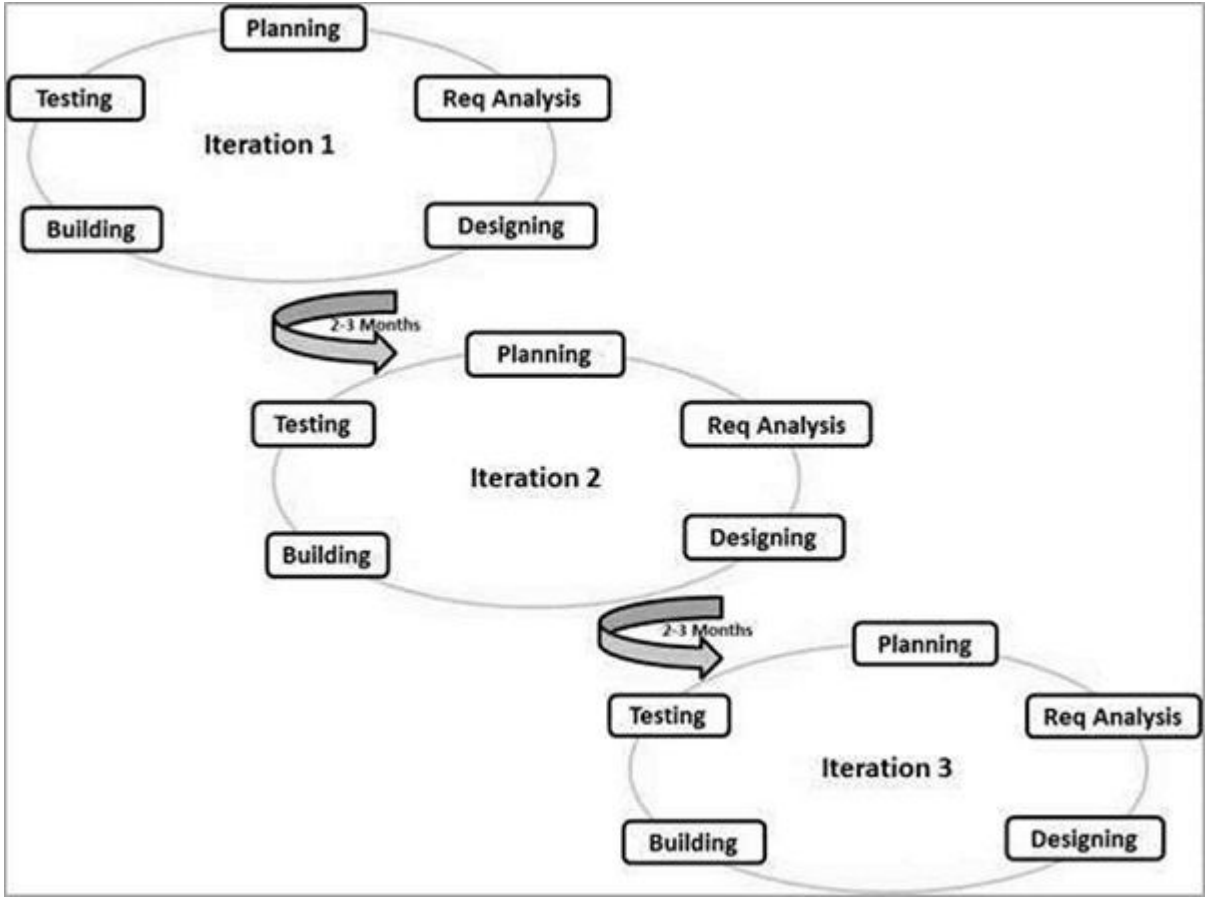
Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In Agile, the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.

Every iteration involves cross functional teams working simultaneously on various areas like −

- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing.

The Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability.



WHY HAVE WE CHOSEN AGILE MODEL ?

Agile uses an adaptive approach where there is no detailed planning and there is clarity on future tasks only in respect of what features need to be developed. There is feature driven development and the team adapts to the changing product requirements dynamically. The product is tested very frequently, through the release iterations, minimizing the risk of any major failures in future.

Customer Interaction is the backbone of this Agile methodology, and open communication with minimum documentation are the typical features of Agile development environment. The agile teams work in close collaboration with each other and are most often located in the same geographical location.

ADVANTAGES:

- Is a very realistic approach to software development.
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.
- Good model for environments that change steadily.

DISADVANTAGES:

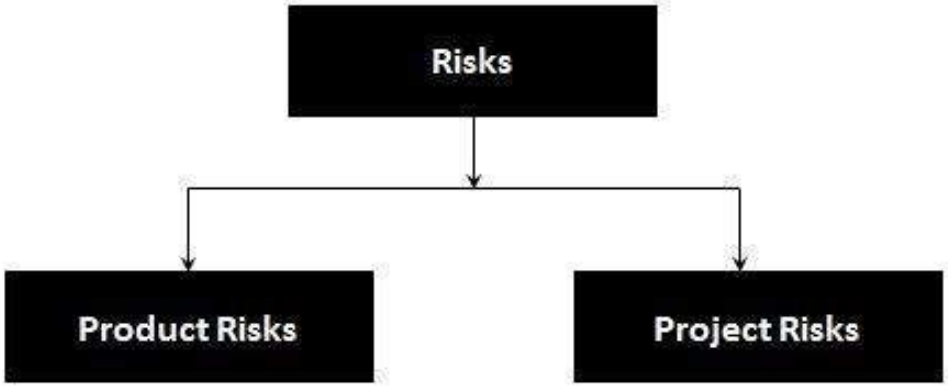
- Not suitable for handling complex dependencies.
- More risk of sustainability, maintainability and extensibility.
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.

WHAT IS RISK?

Risk can be defined as the probability of an event, hazard, accident, threat or situation occurring and its undesirable consequences. It is a factor that could result in negative consequences and usually expressed as the product of impact and likelihood.

Risk = Probability of the event occurring x Impact if it did happen

In software terminology, the risk is broadly divided into two main categories:



PROJECT RISKS

- Supplier Issues
- Organizational factors
- Technical issues

PRODUCT RISKS

- Defect Prone Software delivered
- The Critical defects in the product that could cause harm to an individual (injury or death) or company
- Poor software Features
- Inconsistent Software Features

BUSINESS RISK

Business risk is the exposure a company or organization has that will lower its profits or lead it to fail. Anything that threatens a company's ability to achieve its financial goals is considered a business risk.

TYPES OF RISK IN SOFTWARE DEVELOPMENT

➔ SCHEDULE RISK

The wrong schedule affects development almost immediately. If project tasks and schedules are not addressed properly, the likelihood of project failure is high. Hence, it is important to keep in mind the areas where schedule risk is highly probable:

- Wrong time estimation
- Resources are not tracked properly. All resources like staff, systems, skills of individuals, etc.
- Failure to identify complex functionalities and time required to develop those functionalities.
- Unexpected project scope expansions.

➔ BUDGET RISK

The finance distribution when done properly leads to reasonable use of finances and creates the grounds for project success.

- Wrong budget estimation.
- Cost overruns
- Project scope expansion

➔ OPERATIONAL RISKS

Risks of loss due to improper process implementation failed system or some external events risks. Causes of Operational Risks:

- Failure to address priority conflicts
- Failure to resolve the responsibilities
- Insufficient resources
- No proper subject training
- No resource planning
- No communication in the team.

➔ TECHNICAL RISKS

Technical risks generally lead to failure of functionality and performance. Causes of Technical Risks are:

- Continuous changing requirements
- No advanced technology available or the existing technology is in the initial stages.
- The product is complex to implement.
- Difficult project modules integration.

➔ PROGRAMMATIC RISKS

These are the external risks beyond the operational limits. These are all uncertain risks are outside the control of the program. These external events can be:

- Running out of the fund.
- Market development
- Changing customer product strategy and priority
- Government rule changes.

RISK MITIGATION & MONITORING

Risk Mitigation is a problem avoidance activity, Risk Monitoring is a project tracking activity,Risk Management includes contingency plans that risk will occur.

Monitoring risk, including tracking identified risks and evaluating the performance of risk mitigation actions is critical to the risk mitigation process. Systematically monitoring risk feeds information back into the other risk management activities, such as identification, analysis, mitigation planning, and mitigation plan implementation.

The process for risk monitoring includes setting a structure for how often you review your risk, what to monitor, how to report changes, and how to redefine your risk strategies.

➔ RISK MITIGATION

Related to risk planning, through risk mitigation, the team develops strategies to reduce the possibility or the loss impact of a risk. Risk mitigation produces a situation in which the risk items are eliminated or otherwise resolved.

Five steps for creating a risk mitigation plan:-

- **IDENTIFY THE RISK**- Identify the potential events and event sequences where risk is presented. Risks can be in the form of existing vulnerabilities in the organization or known threats.
- **PERFORM A RISK ASSESSMENT**- Find the quantitative risk of each event by weighing its potential impact and the likelihood of it occurring.
- **PRIORITIZE**- Once the risk assessment has been completed, rank the potential risks from most severe to least. Areas with lowest level of the acceptable risk should be the priority.
- **TRACK RISKS**- If a risk can be followed, keep the track of it and the threat it poses. For example- Monitor the frequency of cyberattacks in your industry.
- **IMPLEMENT & MONITOR PROGRESS**- Once your mitigation plan is in place, continue to monitor how it is working and perform tests to ensure the plan is up to date. If risk priorities change, make sure your plan reflects that.

➔ RISK MONITORING

Project risk control and risk monitoring is where you keep track of about how your risk responses are performing against the plan as well as the place where new risks to the project are managed.

You must remember that risks can have negative and positive impacts. Positive risk is a risk taken by the project because its potential benefits outweigh the traditional approach and a negative risk is one that could negatively influence the cost of the project or its schedule.

A Continuous Interlocked Process—Not an Event

RISK MANAGEMENT

Risk management is the identification, evaluation, and prioritization of risks followed by coordinated and economical application of resources to minimize, monitor, and control the probability or impact of unfortunate events or to maximize the realization of opportunities.

5 risk management process steps combine to deliver a simple and effective risk management process are:-

- **Step 1: Identify the Risk.** Uncover, recognize and describe risks that might affect your project or its outcomes. There are a number of techniques you can use to find project risks. During this step you start to prepare your Project Risk Register.
- **Step 2: Analyze the risk**. Once risks are identified you determine the likelihood and consequence of each risk. You develop an understanding of the nature of the risk and its potential to affect project goals and objectives.
- **Step 3: Evaluate or Rank the Risk.** You evaluate or rank the risk by determining the risk magnitude, which is the combination of likelihood and consequence. You make decisions about whether the risk is acceptable or whether it is serious enough to warrant treatment.
- **Step 4: Treat the Risk.** This is also referred to as Risk Response Planning. During this step you assess your highest ranked risks and set out a plan to treat or modify these risks to achieve acceptable risk levels.You create risk mitigation strategies, preventive plans and contingency plans in this step.
- **Step 5: Monitor and Review the risk.** Risk is about uncertainty. If you put a framework around that uncertainty, then you effectively de-risk your project.And that means you can move much more confidently to achieve your project goals. By identifying and managing a comprehensive list of project risks, unpleasant surprises and barriers can be reduced.


Risk Identification
Risk Analysis
Risk Planning
Risk Monitoring
List of Potential risks
Prioritized risk list
Risk avoidance and contingency plans
Risk Assessment

CONFIGURATION MANAGEMENT

Software development life cycle (SDLC) is a series of phases that provide a common understanding of the software building process. How the software will be realized and developed from the business understanding and requirements elicitation phase to convert these business ideas and requirements into functions and features until its usage and operation to achieve the business needs.

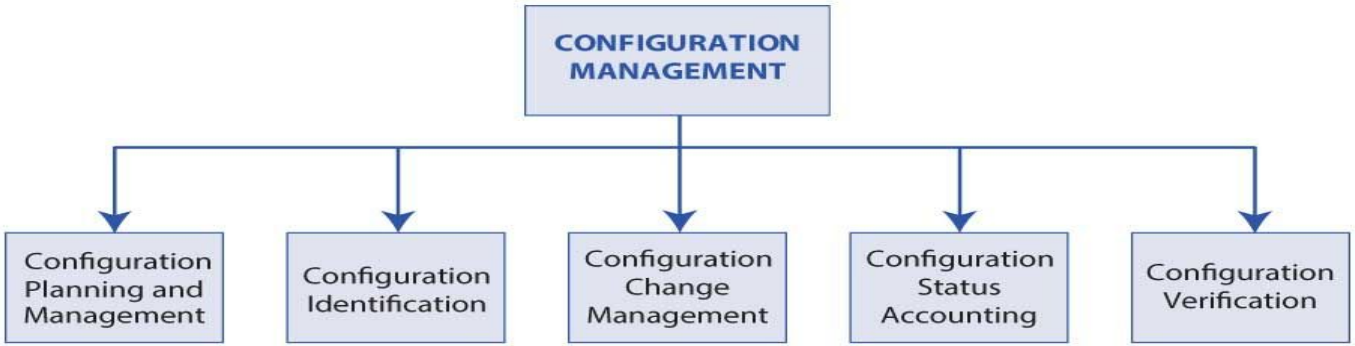


FIGURE 6.5-2 Five Elements of Configuration Management

The following risks could be involved in our model:-

- There could be a **technical** and **programmatic** risks. There could be a need for correction of errors or updation of the code.
- There could also be a **budget risk** due to cost overruns or wrong budget estimation.
- There could be a **risk of targeted attacks**. Smart home device hold a treasure trove of personal information, that cybercriminals can steal via hacking if the device lacks robust protections.
- **Device Hijacking**- The attacker hijacks and effectively assumes control of a device.
- Smart appliance can provide cyber attackers ample amount of **personal information** that can be exploited for fraudulent transactions or theft.
- **Permanent denial of service(PDOS)**, also known as phlashing, is an attack that damages the device so badly that it requires replacement or reinstallation of hardware.

The following things could be done to avoid or risks involved in our model:-

- **SECURE BOOT**- Secure boot utilizes cryptographic code signing techniques, ensuring that a device only executes code generated by the device OEM or another trusted party.
- **MUTUAL AUTHENTICATION**- Every time a smart home device connects to a network, it should be authenticated prior to receiving or transmitting data. This ensures data originates from a legitimate device & not a fraudulent source.
- **SECURE COMMUNICATION(ENCRYPTION)**- Protecting data in transit between a device and its service infrastructure(the cloud).
- **SECURITY MONITORING & ANALYSIS**- Captures data on the overall state of the system, including endpoint devices and connectivity traffic. This data is then analyzed to detect possible security violations or potential system threats.
- **SECURITY LIFECYCLE MANAGEMENT**- The lifecycle management feature allows service providers & OEMs to control the security aspects of the IOT devices when in operation.
- The **technical** and **programmatic** errors can be solved by finding the practical bugs in the code or program.
- **Budget risk** can be avoided by doing a budget planning before making of the project.

SOFTWARE QUALITY METRICS

WHAT IS SOFTWARE QUALITY METRICS?

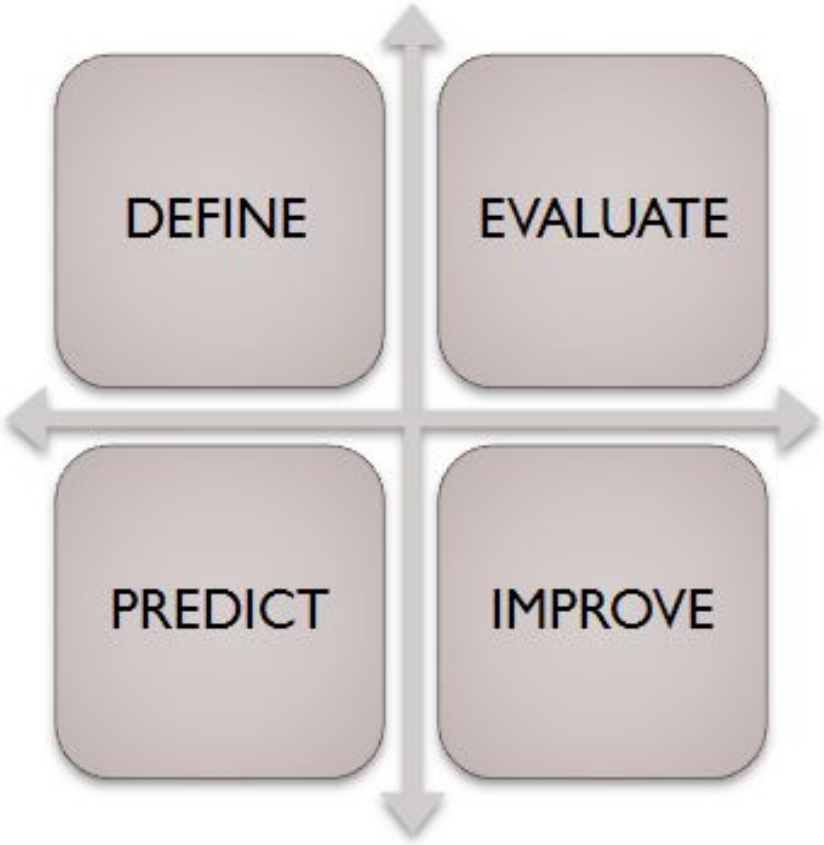
The word '**metrics**' refer to standards for measurements. Software Quality Metrics means measurement of attributes, pertaining to software quality along with its process of development.

The term **"software quality metrics"** illustrate the picture of measuring the software qualities by recording the number of defects or security loopholes present in the software. However, quality measurement is not restricted to counting of defects or vulnerabilities but also covers other aspects of the qualities such as maintainability, reliability, integrity, usability, customer satisfaction, etc.


SOFTWARE
QUALITY
METRICS

WHY SOFTWARE QUALITY METRICS?

- To define and categorize elements in order to have better understanding of each and every process and attribute.
- To evaluate and assess each of these process and attribute against the given requirements and specifications.
- Predicting and planning the next move w.r.t software and business requirements.
- Improving the Overall quality of the process and product, and subsequently of project.


DEFINE
EVALUATE
PREDICT
IMPROVE

PROCESS METRICS

These are metrics that pertain to Process Quality. They are used to measure the efficiency and effectiveness of various processes.

Process metrics are an invaluable tool for companies wanting to monitor, evaluate, and improve their operational performance across the enterprise. But for a combination of reasons, companies often fail to use process metrics to the extent that they could and should.

- **Cost of quality: It is a measure of the performance of quality initiatives in an organization. It's expressed in monetary terms.**
  Cost of quality = (review + testing + verification review + verification testing + QA + configuration management + measurement + training + rework review + rework testing)/ total effort x 100.
- **Cost of poor quality: It is the cost of implementing imperfect processes and products.**
  Cost of poor quality = rework effort/ total effort x 100.
- **Defect density: It is the number of defects detected in the software during development divided by the size of the software (typically in KLOC or FP)**
  Defect density for a project = Total number of defects/ project size in KLOC or FP
- **Review efficiency: defined as the efficiency in harnessing/ detecting review defects in the verification stage.**
  Review efficiency = (number of defects caught in review)/ total number of defects caught) x 100.

- **Testing Efficiency: Testing efficiency = 1 – ((defects found in acceptance)/ total number of testing defects) x 100.**
- **Defect removal efficiency: Quantifies the efficiency with which defects were detected and prevented from reaching the customer.**
  Defect removal efficiency = (1 – (total defects caught by customer/ total number of defects)) x 100.
- **Residual defect density = (total number of defects found by a customer)/ (Total number of defects including customer found defects) x 100.**

PROJECT METRICS

These are the metrics pertaining to the Project Quality. They measure defects, cost, schedule, productivity and estimation of various project resources and deliverables.

- It may include the number of teams, developers involved, cost and duration for the project, etc.

Some metrics belong to multiple categories. For example, the in-process quality metrics of a project are both process metrics and project metrics.

Project metrics enable software manager to:-

- Minimize the development time by making the adjustments necessary to avoid delays and potential problems and risks.
- Assess product quality on an ongoing basis & modify the technical approach to improve quality.
- Used in estimation techniques & other technical work.

- Metrics collected from past projects are used as a basis from which effort and time estimates are made for current software project.
- As a project proceeds, actual values of human effort & calendar time expended are compared to the original estimates.
- This data is used by the project manager to monitor & control the project.

PRODUCT METRICS

- Product metrics are measures of the software product at any stage of its development, from requirements to installed system. It describes the characteristics of the product such as size, complexity, design features, performance, and quality level. Product metrics may measure:
  - the complexity of the software design
  - the size of the final program
  - the number of pages of documentation produced

This metrics include the following −

- Mean Time to Failure
- Defect Density
- Customer Problems
- Customer Satisfaction

MEAN TIME TO FAILURE

It is the time between failures. This metric is mostly used with safety critical systems such as the airline traffic control systems, avionics, and weapons.

DEFECT DENSITY

It measures the defects relative to the software size expressed as lines of code or function point, etc. i.e., it measures code quality per unit. This metric is used in many commercial software systems.

CUSTOMER PROBLEMS

It measures the problems that customers encounter when using the product. It contains the customer's perspective towards the problem space of the software, which includes the non-defect oriented problems together with the defect problems.

The problems metric is usually expressed in terms of Problems per User-Month (PUM).

CUSTOMER SATISFACTION

Customer satisfaction is often measured by customer survey data through the five-point scale −

- Mean Time to Failure
- Defect Density
- Customer Problems
- Customer Satisfaction
- Very satisfied
- Satisfied
- Neutral
- Dissatisfied
- Very dissatisfied

Satisfaction with the overall quality of the product and its specific dimensions is usually obtained through various methods of customer surveys. Based on the five-point-scale data, several metrics with slight variations can be constructed and used, depending on the purpose of analysis.

Software Quality Metrics wrt Our Model

Product Metrics: Our model is a very compact model which can be installed inside any switchboard or just outside switchboard for easy maintenance. The lights indicate the switching or path of current flow.

Process Metrics: This model can be re-modified or updated anytime one wants to. External installation helps in doing so. The working model is running with zero error but has a small glitch that is it has a latency.

Project Metrics: The model includes 3 developers, for arduino coding of three different showcases. It's not too costly only an investment of nearly ₹1000/- is enough for one to two home equipments. This model is useful for electricity cost cutting and conservation

THE END