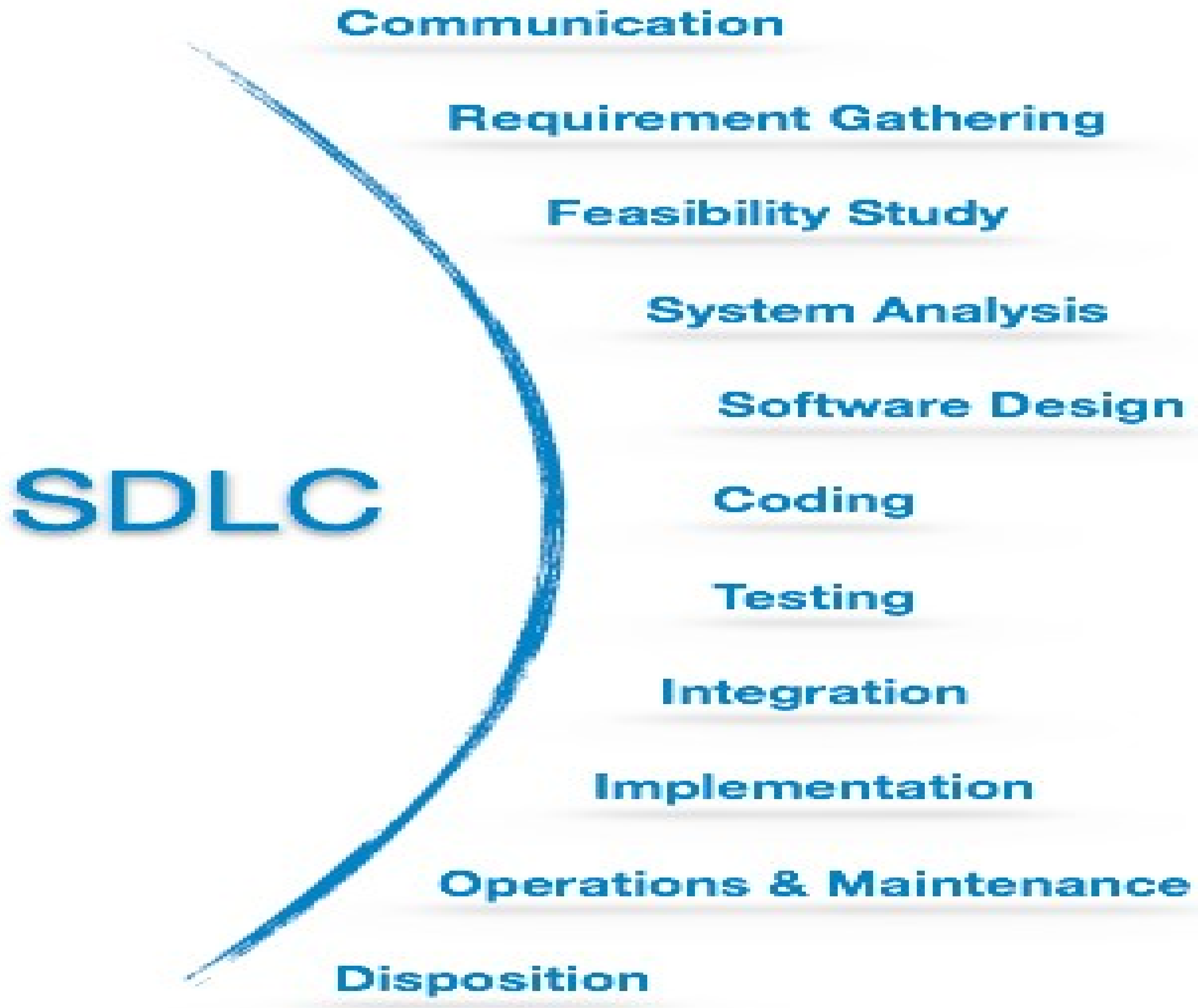
Software Development Life Cycle (SDLC)

Software Development Life Cycle (SDLC)

- It is the division of software development work into distinct phases (or stages) containing activities with the intent of better planning and management.
- SDLC for short, is a well-defined, structured sequence of stages in software engineering to develop the intended software product.
- SDLC provides a series of steps to be followed to design and develop a software product efficiently. SDLC framework includes the following steps which are shown in the next slide:


SDLC
Communication
Requirement Gathering
Feasibility Study
System Analysis
Software Design
Coding
Testing
Integration
Implementation
Operations & Maintenance
Disposition

Communication

- This is the first step where the user initiates the request for a desired software product.
- He contacts the service provider and tries to negotiate the terms.
- He submits his request to the service providing organization in writing.

Requirement Gathering

- This step onwards the software development team works to carry on the project.
- The team holds discussions with various stakeholders from problem domain and tries to bring out as much information as possible on their requirements.
- The requirements are contemplated and segregated into

  -user requirements,

  -system requirements

  -functional requirements,

  -non-functional requirements.

Requirement Gathering

Functional Requirements :

- We develop projects/software to meet Functional requirements.
- These requirements are the desired functionality that the client want us to build and delivered to them.
- A functional requirement describes an interaction between the system and its environment.
- Let’s just say, decide the inputs to the system and output from the system considering all internal and external factors involved.
- A document is created logging all details which contains what a certain system has to do to achieve a certain specific objective.

Requirement Gathering

Non-functional requirements:

- Non-functional requirements are the untold part of the project which are not communicated but readily understood as a Global standard.
- These can be understood as a support system of functional requirements.
- These are the expectations of client to showcase the product as a global product following global guidelines.

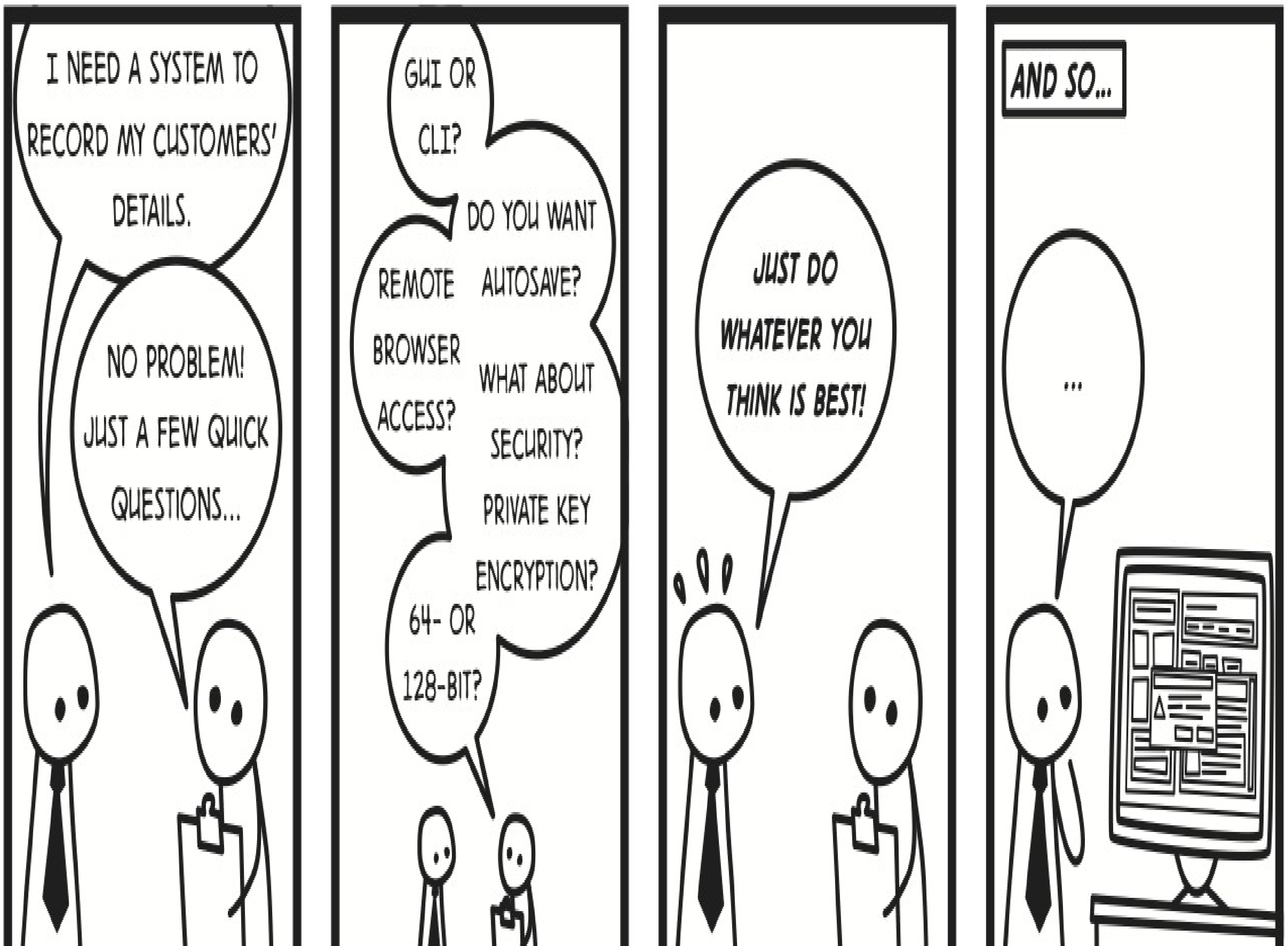
Requirement Gathering

- A nonfunctional requirement can be described as a constraint on the system that like page load speed which can be optimized up-to a certain extent.
- Non functional requirements can be limited to Response time, security, reliability, accuracy, capacity and availability. Clients has to decide on all these parameters and take a call to achieve a workable.

Requirement Gathering

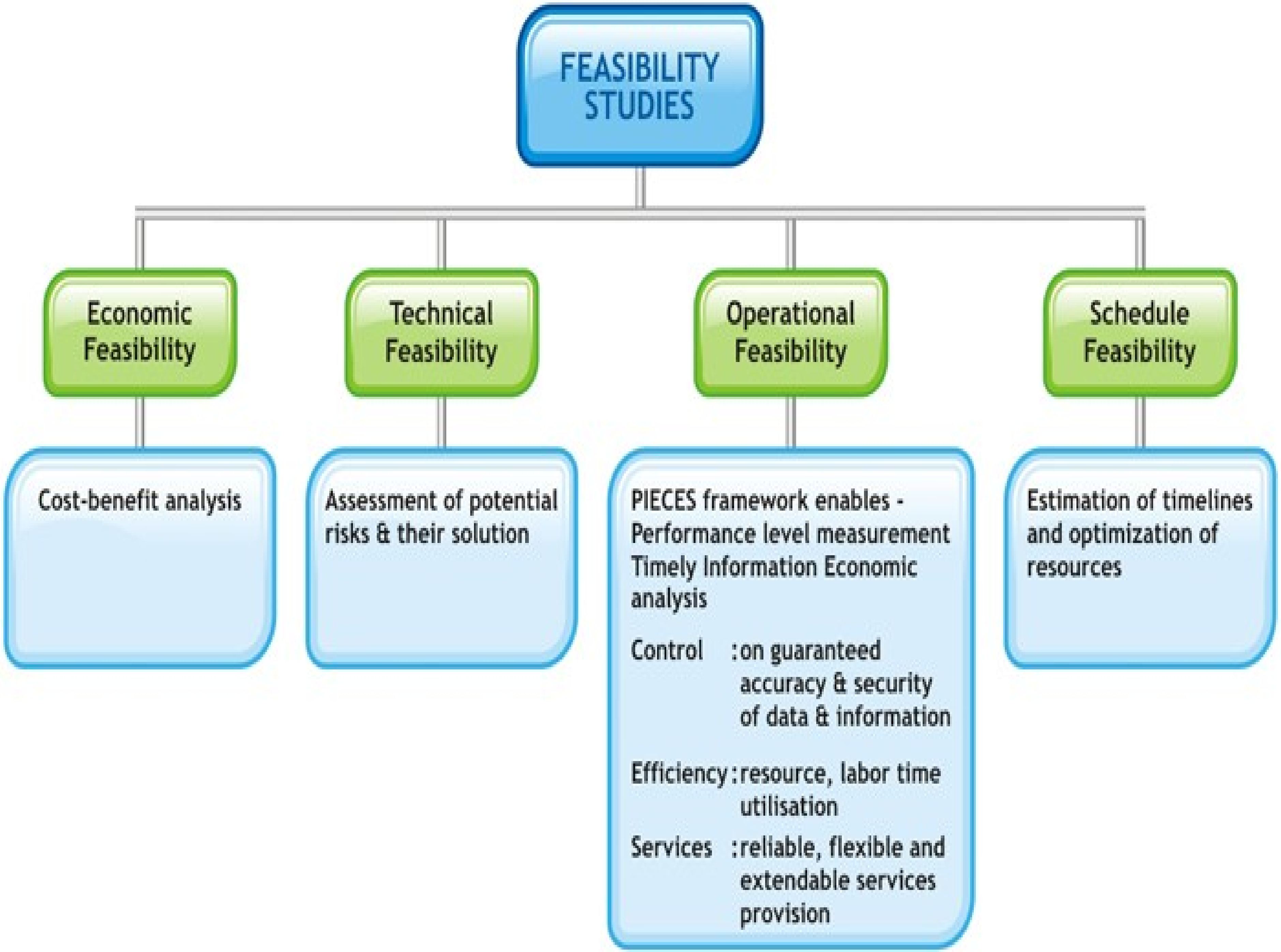
The requirements are collected using a number of practices as given -

- Studying the existing or obsolete system and software,
- Conducting interviews of users and developers.
- Survey is the good technique to collect requirements within short period of time. In this technique it is necessary to decide goal of survey and then frame questions list.
- Referring to the database or collecting answers from the questionnaires.


I NEED A SYSTEM TO RECORD MY CUSTOMERS' DETAILS.
NO PROBLEM! JUST A FEW QUICK QUESTIONS...
GUI OR CLI?
DO YOU WANT AUTOSAVE?
REMOTE BROWSER ACCESS?
WHAT ABOUT SECURITY?
PRIVATE KEY ENCRYPTION?
64- OR 128-BIT?
JUST DO WHATEVER YOU THINK IS BEST!
AND SO...
...

Feasibility Study

- After requirement gathering, the team comes up with a rough plan of software process.
- At this step the team analyzes if a software can be made to fulfill all requirements of the user and if there is any possibility of software being no more useful.
- It is found out, if the project is financially, practically and technologically feasible for the organization to take up. There are many algorithms available, which help the developers to conclude the feasibility of a software project.


FEASIBILITY STUDIES
Economic Feasibility
Technical Feasibility
Operational Feasibility
Schedule Feasibility
Cost-benefit analysis
Assessment of potential risks & their solution
PIECES framework enables - Performance level measurement Timely Information Economic analysis
Control :on guaranteed accuracy & security of data & information
Efficiency :resource, labor time utilisation
Services :reliable, flexible and extendable services provision
Estimation of timelines and optimization of resources

Feasibility Study

Advantages of making Feasibility study:

- This study being made as the initial step of software development life cycle has all the analysis part in it which helps in analyzing the system requirements completely.
- Helps in identifying the risk factors involved in developing and deploying the system.
- The feasibility study helps in planning for risk analysis.
- Feasibility study helps in making cost/benefit analysis which helps the organization and system to run efficiently.

Feasibility Study

- Feasibility study helps in making plans for training developers for implementing the system.
- So a feasibility study is a report which could be used by the senior or top persons in the organization. This is because based on the report the organization decides about cost estimation, funding and other important decisions which is very essential for an organization to run profitably and for the system to run stable.

System Analysis

- At this step the developers decide a roadmap of their plan and try to bring up the best software model suitable for the project.
- System analysis includes understanding of software product limitations, learning system related problems or changes to be done in existing systems beforehand, identifying and addressing the impact of project on organization and personnel etc.
- The project team analyzes the scope of the project and plans the schedule and resources accordingly.

Design

- Next step is to bring down whole knowledge of requirements and analysis on the desk and design the software product.
- The inputs from users and information gathered in requirement gathering phase are the inputs of this step.
- The output of this step comes in the form of two designs-

  - logical design, and

  -physical design.
- Engineers produce meta-data and data dictionaries, logical diagrams, data-flow diagrams, and in some cases pseudo codes.

Coding

- This step is also known as programming phase.
- The implementation of software design starts in terms of writing program code in the suitable programming language and developing error-free executable programs efficiently.

Testing

- An estimate says that 50% of whole software development process should be tested.
- Errors may ruin the software from critical level to its own removal.
- Software testing is done while coding by the developers and thorough testing is conducted by testing experts at various levels of code such as module testing, program testing, product testing, in-house testing and testing the product at user’s end.
- Early discovery of errors and their remedy is the key to reliable software.

Integration

- Software may need to be integrated with the libraries, databases and other program(s).
- This stage of SDLC is involved in the integration of software with outer world entities.

Implementation

- This means installing the software on user machines. At times, software needs post-installation configurations at user end.
- Software is tested for portability and adaptability and integration related issues are solved during implementation.

Operation and Maintenance

- This phase confirms the software operation in terms of more efficiency and less errors. If required, the users are trained on, or aided with the documentation on how to operate the software and how to keep the software operational.
- The software is maintained timely by updating the code according to the changes taking place in user end environment or technology.
- This phase may face challenges from hidden bugs and real-world unidentified problems.

Disposition

- As time elapses, the software may decline on the performance front. It may go completely obsolete or may need intense upgradation. Hence a pressing need to eliminate a major portion of the system arises.
- This phase includes archiving data and required software components, closing down the system, planning disposition activity and terminating system at appropriate end-of-system time.