UNIT–1 : Introduction & SDLC Models (PYQ Solution)

AKTU · Software Engineering (BCS601) · Solutions by **Chirag Jain**, Ycotes

SECTION A Answers (≈70 Words Each)

Q1. Differentiate between software and program giving various components of software. (May 2026, May 2025, 2023)

A program is just a set of instructions (source or executable code) written to perform one task, usually by its author alone. Software is the complete product: programs plus the documentation and data needed to install, use and maintain them, built by a team and sold or supported for years. The components of software are: programs (instructions), data structures (data the programs use), documentation (SRS, design, user manual, test plans) and operating procedures (installation, backup, help-desk instructions).

Q2. Define generic software with example. (2023)

Generic software is a product developed by a company for the open market and sold to many customers with common needs. The developer owns the specification and decides its features and changes. Examples: Microsoft Word, Adobe Photoshop, Tally, antivirus programs and operating systems. It is opposite to custom (bespoke) software, which is built for one customer against that customer's own specification.

Q3. Justify the statement "Software does not wear out and is not manufactured". (May 2025)

Hardware fails from physical ageing (heat, friction, corrosion), so its failure curve is a bathtub curve. Software has no physical parts, so it does not wear out; its defects are design defects present from the start. However, every change adds new defects, so the failure rate rises after each update and the "idealised curve" turns into a jagged curve. Software is developed, not

manufactured: the cost lies in design and engineering, and producing copies costs almost nothing, unlike a factory product.

Q4. What is meant by software prototyping? (2023)

Software prototyping is building a quick, partial working model of the system to understand and validate the requirements before the real system is developed. The developer collects basic needs, builds a "quick design" prototype, the customer evaluates it, and the prototype is refined until requirements are clear. It reduces misunderstanding and risk, particularly when requirements are vague. The prototype is usually thrown away (throw-away prototyping) or evolved into the final product (evolutionary prototyping).

Q5. Mention the advantages of CASE tools. (2023)

CASE (Computer-Aided Software Engineering) tools automate software development activities. Their advantages are: higher productivity and faster development through automation of repetitive work; better quality because of automatic error and consistency checking; improved documentation generated automatically from the design; easier maintenance and change control; standardisation of notations and methods; and better communication among team members through a shared repository.

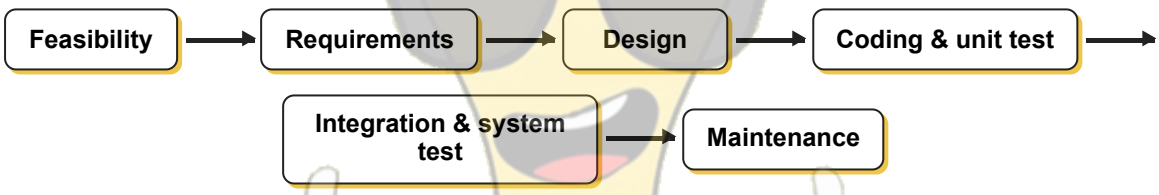
SECTION B Answers (≈250 Words Each)

Q1. Iterative waterfall and spiral model for software lifecycle clean and discuss various activities in each phase. (2023)

Both are **software life cycle models**: the iterative waterfall is a linear model with feedback, the spiral is a risk-driven model.

Iterative waterfall model

- **Feasibility study** – decide if the project is financially and technically viable.
- **Requirements analysis and specification** – gather needs and write the SRS.
- **Design** – architecture and detailed design of modules.
- **Coding and unit testing** – implement and test each module.
- **Integration and system testing** – combine modules and test the whole system.
- **Maintenance** – corrective, adaptive and perfective changes.
- Feedback paths let the team return to an earlier phase to correct errors, unlike the classical waterfall.



Iterative waterfall model (with feedback to earlier phases)

Spiral model

Each loop of the spiral has four quadrants:

1. **Determine objectives**, alternatives and constraints.
2. **Identify and resolve risks** (prototyping, simulation).
3. **Develop and verify** the product of that cycle.
4. **Plan the next cycle** with customer review.

The radius shows cost and the angle shows progress. Loops are typically concept, requirements, design and construction.

Point	Iterative waterfall	Spiral
Nature	Linear, feedback	Risk-driven, iterative
Risk handling	Weak	Explicit in every loop
Customer feedback	At the end	After each loop
Best for	Well-understood requirements	Large, risky projects

Conclusion: the iterative waterfall is simple and easy to manage, whereas the spiral model is better for large and risky projects.

Q2. What are the problems faced in software engineering? Explain each in detail. (2023)

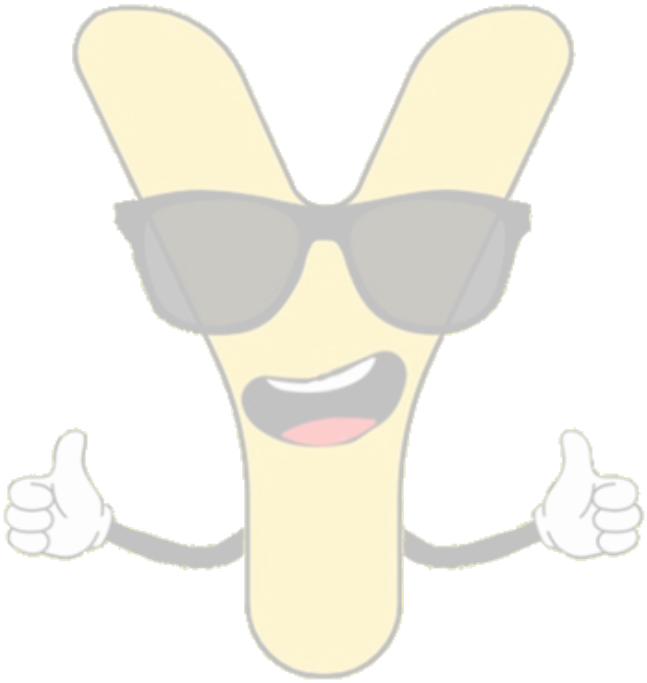
Software engineering has to deal with several recurring problems, which together led to the "software crisis".

- **Cost and schedule overruns** – estimates are made with little data, so most projects exceed budget and deadline. Poor tracking and changing scope make it worse.
- **Changing and unclear requirements** – customers cannot state everything at the start and the needs change; unclear requirements lead to rework or the wrong product.
- **Poor quality and reliability** – defects remain after delivery because testing is incomplete, causing failures and loss of user trust.
- **Complexity and scale** – large systems have thousands of interacting modules; no one person understands everything, so integration problems arise.
- **Maintenance difficulty** – badly structured, undocumented code is hard to understand and change; maintenance takes 60-70% of total cost.
- **Poor project management and communication** – lack of planning, risk management and team coordination causes delays and conflicts.
- **Shortage of skilled people and low productivity** – demand for developers is higher than supply, and productivity is hard to measure.
- **Lack of reuse and standards** – every project starts from scratch, repeating effort and mistakes.
- **Rapid technology change** – tools and platforms become obsolete quickly, forcing retraining and redesign.

- **Intangibility** – software cannot be seen or touched, so progress is hard to measure.

Remedies: use a suitable life cycle model, standards, reviews, CASE tools, risk management, reuse and good project management.

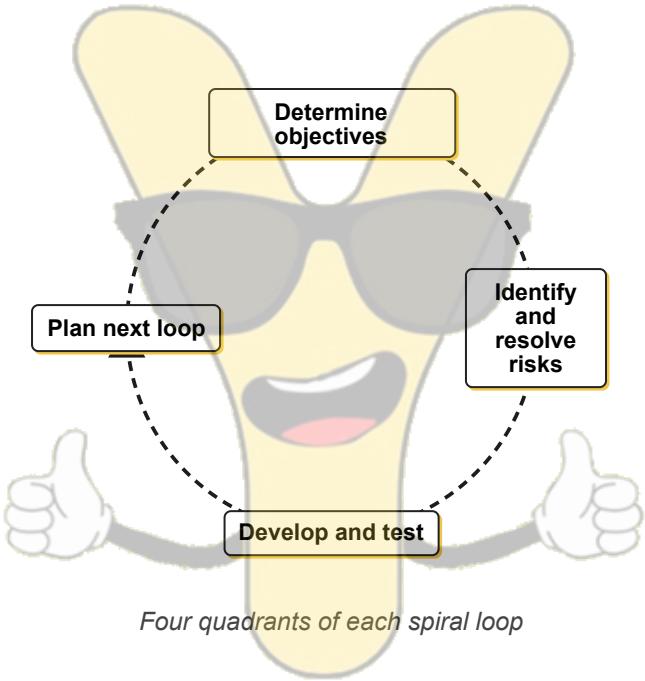
Conclusion: these problems arise because software is complex and invisible, and software engineering applies systematic, disciplined methods to control them.



SECTION C Answers (≈500 Words Each)

Q1. Discuss the Spiral Model and justify that this model incorporates good features of almost every other model i.e. Water Fall Model, Prototype model, Iterative enhancement models etc. Also discuss the criteria for selection of model for software development. (May 2026, May 2025)

The **Spiral Model**, proposed by Barry Boehm (1988), is a **risk-driven, evolutionary** process model. The project passes through a number of loops of a spiral; each loop is one phase of development (feasibility, requirements, design, construction). The radius of the spiral shows the cost accumulated so far and the angle shows progress.



Four quadrants of each spiral loop

Four quadrants of every loop

1. **Objective setting** – define objectives, alternatives and constraints of this phase.
2. **Risk assessment and reduction** – analyse risks and resolve them by prototyping, simulation or benchmarking.
3. **Development and validation** – build and verify the product for that loop.
4. **Planning** – review with the customer and plan the next loop.

Advantages: risk handling, early customer feedback, suited to large and complex projects, change accepted. **Disadvantages:** needs risk-analysis expertise, costly, hard to manage for small projects, no fixed end.

Justification that spiral absorbs other models

- **Waterfall:** the sequence requirements → design → code → test appears in each loop in a controlled way; a one-loop spiral with no risks reduces to waterfall.
- **Prototype model:** the second quadrant builds prototypes to resolve risk, so prototyping is built in.
- **Iterative enhancement / incremental:** every loop produces a more complete version, which is incremental delivery.
- **Evolutionary model:** the product grows over loops and changes are welcomed.
- **Risk-driven plus customer review:** it adds risk analysis and a review at the end of every loop, which the others lack.
- Therefore it is called a **meta-model**, because it can emulate the other models.

Criteria for selection of a life cycle model

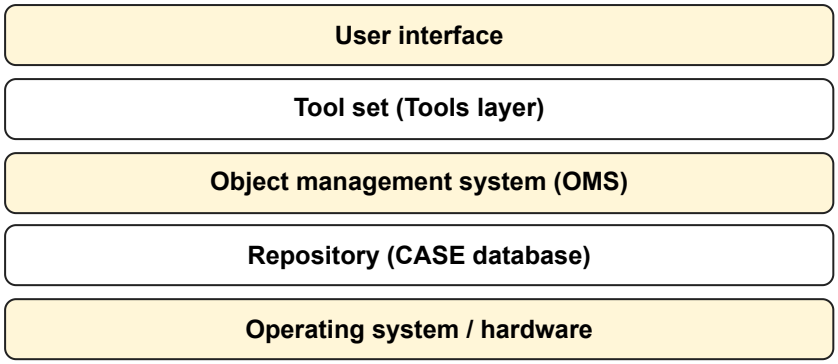
Criterion	Choose
Requirements clear and stable	Waterfall
Requirements unclear	Prototype
High risk, large project	Spiral
Early partial delivery needed	Incremental / RAD
Frequent change, close customer	Agile

Other factors are team skill, project size, schedule, budget, technology, documentation needs and customer involvement.

Conclusion: the spiral model combines the systematic steps of waterfall, the prototyping of the prototype model and the incremental growth of iterative models with explicit risk management, so it suits large, risky projects.

Q2. Give the schematic representation of CASE environment architecture. Give a brief description of its components. What are various benefits of using CASE tools? (May 2026, May 2025)

CASE (Computer-Aided Software Engineering) tools are software programs that support activities of the software life cycle such as analysis, design, coding, testing and maintenance. A **CASE environment** is the integrated set of tools that share information through a common repository.



CASE environment architecture

Components

- **User interface** – a consistent window-and-menu interface through which all tools are used, with a common look and feel.
- **Tool set (tools layer)** – individual tools such as diagram editors, analysers, code generators, test tools and project-management tools; upper-CASE (analysis and design) and lower-CASE (coding and testing).
- **Object management system (OMS)** – handles the storage, retrieval and consistency of the objects (configuration items) kept in the repository; supports version control.
- **Repository (CASE database)** – a central database that stores all project information: requirements, designs, code, test data, documents and the relationships between them.
- **Operating system and hardware** – the underlying platform, often a network supporting multiple users.
- **Integration framework** – lets tools share data and be invoked together (data, control, presentation integration).

Types of CASE tools: upper CASE (requirement and design tools), lower CASE (code generators, testing tools) and integrated CASE (I-CASE) which covers the whole life cycle.

Benefits of CASE tools

- **Higher productivity** because repetitive work is automated.
- **Better quality**, since design rules and consistency are checked automatically.
- **Faster development** and shorter delivery time.
- **Automatic documentation** that stays consistent with the design.
- **Improved communication** and teamwork through the shared repository.
- **Easier maintenance** through traceability and change control.
- **Standardisation** of methods and notations.

- **Reuse** of design and code components.
- **Easy re-engineering** and prototyping support.

Limitations: cost of tools and training, learning curve, over-dependence on tools.

Conclusion: a CASE environment integrates tools around a repository through the OMS and the user interface, and it improves productivity, quality and maintainability of the software.

Q3. Define software and software engineering. Explain software characteristics, write various components of software and give their brief description. Why is there a crisis in software development industry? (May 2025, 2023)

Software is a collection of computer programs together with the associated data, documentation and procedures that perform a required function. **Software engineering** is the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software (IEEE).

Characteristics of software

- **Developed, not manufactured** – cost lies in design; copies are nearly free.
- **Does not wear out** – no physical ageing, but defects accumulate with changes (idealised vs actual failure curve).
- **Custom built** – most software is made for a specific need rather than assembled from standard parts, although component reuse is increasing.
- **Intangible** – cannot be seen or touched, so progress is hard to measure.
- **Complex and flexible** – easily changed but with side effects.
- **Quality attributes** – reliability, efficiency, maintainability, usability.

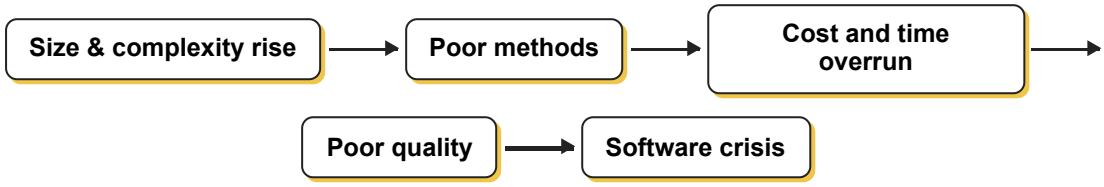
Components of software

- **Programs** – instructions that run on a computer.
- **Data structures** – data and files the programs manipulate.
- **Documentation** – SRS, design documents, test plans, user and installation manuals.
- **Operating procedures** – setup, backup and operating instructions.

Software crisis – reasons

- **Cost and schedule overruns** due to poor estimation.

- **Poor quality** – many defects and low reliability.
- **Changing and unclear requirements.**
- **Growth in size and complexity** of software.
- **Inadequate maintainability** and poor documentation.
- **Lack of trained people**, tools and standard methods; low productivity.
- **Poor project management** and lack of early planning and risk handling.
- **Rapid change in technology and hardware**.



How the software crisis arises

Solution: the crisis led to the creation of software engineering, using life cycle models, standards, CASE tools, metrics, reviews and good project management.

Conclusion: software is more than program code, its special characteristics make it hard to build, and the crisis in the industry is controlled by applying engineering principles.

