Software Engineering IA-1 Solutions (GPT)

Parth R.

Q1A) Answer the following: A software development firm decides to develop a new operating system. You being the software developer, analyse the software development process and framework, and distinguish between them.

Answer:

When a software development firm decides to create a new operating system, it must carefully plan both the **software development process** and the **framework** that will guide the work.

1. Software Development Process

The **process** is the set of ordered steps and phases followed to develop the software. It defines **what needs to be done** and in which sequence. The main stages include:

1. **Requirement Analysis** – Collecting needs from stakeholders, such as security requirements, memory management, device driver support, etc.
2. **System Design** – Creating architectural models for the OS kernel, file system, process management, and user interface.
3. **Implementation (Coding)** – Writing the core OS modules in low-level languages such as C, C++, or Assembly.
4. **Testing** – Unit testing, integration testing, and system testing to ensure correctness, efficiency, and reliability.
5. **Deployment** – Releasing the OS for installation on hardware.
6. **Maintenance and Updates** – Fixing bugs, providing patches, and releasing upgrades.

Examples of software development processes:

- **Waterfall Model** – Linear, step-by-step approach.
- **Spiral Model** – Combines iterative development with risk analysis.
- **Agile Process** – Incremental development with frequent feedback.
- **V-Model** – Emphasizes testing at each stage of development.

Thus, the process provides a *roadmap* for creating the operating system.

2. Software Development Framework

A **framework** provides the practical **methodologies, tools, standards, and practices** that help implement the chosen process. It defines **how the work should be carried out**. Frameworks often specify roles, responsibilities, workflows, and artifacts.

For example:

- If the firm follows **Agile**, then the framework could be **Scrum**, which defines sprint planning, daily stand-up meetings, product backlogs, and sprint reviews.
- If the firm chooses a large-scale development model, it might use the **Rational Unified Process (RUP)**, which divides the work into phases like inception, elaboration, construction, and transition.
- Other frameworks like **Extreme Programming (XP)** stress practices such as pair programming, test-driven development, and continuous integration.

So, while the process says *"we will follow Agile"*, the framework says *"we will follow Scrum with 2-week sprints, sprint retrospectives, and backlog management."*

---

3. Key Distinction

Aspect	Software Development Process	Software Development Framework
Definition	Sequence of steps/phases to develop software	Structured methodology, tools, and practices to implement the process
Focus	*What to do*	*How to do it*
Examples	Waterfall, Spiral, Agile, V-Model	Scrum, RUP, XP, SAFe
Role in OS Development	Ensures systematic stages from requirement analysis to maintenance	Defines collaboration, iteration, and quality assurance methods

Conclusion

For developing a new operating system:

- The **process** ensures systematic progress—starting from gathering requirements to maintenance.

- The **framework** ensures the team collaborates efficiently, uses best practices, and delivers the OS in manageable increments.

Q1B1) A group of software developers have been given the task analyze the approach and techniques followed by an organization. You being the member of the group, scrutinize the software development process of the organization and appraise the management of the organization about the various levels of the capability maturity model that the organization should follow to improve the process.

Answer:

As a member of the group analyzing the organization's software development process, it is important to evaluate the maturity of their practices and recommend structured improvements using the **Capability Maturity Model (CMM)**.

The **CMM** is a framework developed to assess and improve software development processes within an organization. It defines five maturity levels, each representing a stage of process capability and discipline. By following these levels, an organization can progressively improve quality, efficiency, and predictability of its software projects.

Levels of the Capability Maturity Model (CMM):

1. **Level 1 – Initial (Ad hoc / Chaotic)**
    - Processes are unpredictable, poorly controlled, and reactive.
    - Success depends on individual effort rather than organizational practices.
    - Example: Developers directly start coding without formal requirements or testing.
    - **Recommendation:** Move towards defined processes instead of relying on individual heroics.
2. **Level 2 – Repeatable (Managed)**
    - Basic project management practices are introduced.
    - Cost, schedule, and functionality can be tracked.
    - Requirements are documented, and past successes can be repeated in similar projects.
    - Example: Maintaining project schedules, using version control systems, and basic testing.
    - **Recommendation:** Standardize project planning and monitoring across teams.

3. **Level 3 – Defined**
    - Processes are documented, standardized, and integrated into a standard organization-wide software process.
    - Training programs are provided so everyone follows the same practices.
    - Example: Organization defines a standard SDLC model (Agile, Spiral, etc.) and mandates its use.
    - **Recommendation:** Establish an organizational process group and invest in process documentation.
4. **Level 4 – Managed**
    - Processes are measured and controlled using quantitative metrics.
    - Software quality and process performance are tracked statistically.
    - Example: Using defect density, code quality metrics, productivity measures, and performance monitoring.
    - **Recommendation:** Implement measurement and analysis techniques to track improvements.
5. **Level 5 – Optimizing**
    - Focus shifts to continuous process improvement.
    - The organization proactively identifies weaknesses and adopts new tools and techniques to improve efficiency.
    - Example: Regular process audits, root-cause analysis, use of AI/automation for testing, continuous feedback loops.
    - **Recommendation:** Encourage innovation, knowledge sharing, and continuous improvement culture.

Appraisal for the Management

- If the organization is at **Level 1 (Initial)**, management must invest in establishing **repeatable practices** such as project tracking and requirement documentation.

- If already at **Level 2 (Repeatable)**, the focus should be on **organization-wide standardization** of processes (Level 3).

- Moving from **Level 3 to 4** requires strong emphasis on **quantitative management**, meaning data-driven decision-making.

- Finally, at **Level 5**, the organization should embrace **continuous improvement and innovation**, ensuring sustained competitiveness in the market.

Conclusion

The Capability Maturity Model provides a roadmap for improving software development processes. By progressing step by step through its levels, the organization can transition from **ad-hoc development** to **optimized, world-class software practices**, ensuring higher productivity, reduced defects, and greater customer satisfaction.

Q1B2) A software development firm is developing a new word processing 04 software. The customer is suggesting changes to the delivered software after working with it. Based on the analysis of the above scenario, relate it to an appropriate software development process model which can be adopted for development of project and also apprise your answer to the management of the firm..

Answer:

In the given scenario, a software development firm is building a **word processing software**. After initial delivery, the customer is suggesting **changes and modifications** based on their experience using the product. This situation clearly shows that the requirements were **not completely known or stable** at the beginning, and the system needs **frequent feedback and refinement**.

Appropriate Software Development Process Model:

The most suitable process model for this project is the **Incremental / Iterative Model** or **Spiral Model**, both of which support customer feedback and progressive improvement.

1. **Incremental Model**
    - The product is developed in **increments** (modules or versions).
    - Each increment delivers a working version of the software with core features (e.g., text editing, saving, printing).
    - Based on user feedback, new increments are developed to add enhancements (spell-check, formatting tools, templates, collaboration features, etc.).
    - **Advantage:** Customers can use early versions, give feedback, and influence further development.

2. **Spiral Model (Risk-driven Iterative Model)**
    - Combines iterative development with **risk analysis**.
    - In each cycle (spiral), requirements are analyzed, prototypes are built, risks are evaluated, and refinements are made.
    - Particularly useful for large projects with **uncertain or changing requirements**.
    - **Advantage:** Strong focus on user feedback, prototyping, and risk management.

Appraisal to Management:

- Since the customer is suggesting changes after delivery, adopting a **linear model like Waterfall** would be **ineffective**, as it assumes fixed requirements and makes changes costly.
- The **Incremental Model** is ideal for word processing software because:
    - It allows delivery of a **basic working product quickly**.
    - Enhancements can be added in future increments as per customer needs.
    - Customer satisfaction improves since feedback is continuously incorporated.
- Alternatively, for large-scale, complex word processors (like MS Word with advanced collaboration features), the **Spiral Model** is beneficial due to its **iterative refinement and risk management**.

Conclusion:

The scenario of customers requesting modifications after delivery aligns with an **Iterative/Incremental or Spiral Model** of software development. These models enable flexibility, incorporate user feedback effectively, and ensure the final word processing software meets customer expectations while reducing risk and cost of rework.

Q2A) A food restaurant management have decided to go online for performing all of their activities. The food restaurant management has given the project to your software organization to develop the project, You being the employee of the software organization has been tasked by the management to identify the requirements of the restaurant management. Analyze the requirement of the restaurant using the requirement engineering process and appraise it to the management.

Answer:

The food restaurant management has decided to take its operations online. As an employee of the software organization, my task is to identify and analyze the requirements using the **Requirement Engineering Process (REP)** and present them to management.

Requirement Engineering Process (REP):

Requirement Engineering is a structured process to gather, analyze, specify, and validate requirements for software development. It involves the following steps:

1. **Requirement Elicitation (Gathering Requirements)**
    - Identify stakeholders (restaurant owner, managers, staff, delivery partners, customers).
    - Techniques used: Interviews, questionnaires, observation, and document study.
    - Collected requirements:
        - Online menu display with categories (starters, main course, desserts, beverages).
        - Online ordering system with customization (add-ons, toppings, spice level).
        - Table reservation system.
        - Online payment integration (UPI, cards, wallets, COD).
        - Delivery tracking and order status updates.
        - Admin panel for restaurant to manage orders, menus, pricing, and discounts.
        - Customer account creation, login, and feedback system.

2. **Requirement Analysis**
    - Classify requirements into **Functional** and **Non-Functional**:
        - **Functional Requirements:** Online order placement, payment processing, reservation booking, notifications, inventory updates, customer reviews.
        - **Non-Functional Requirements:** System scalability (handle peak orders), security (data protection, payment safety), performance (fast response time), reliability (99% uptime).
    - Resolve conflicts: e.g., customer wants COD, but management prefers only prepaid → compromise by offering both.
3. **Requirement Specification (SRS Document)**
    - Requirements are documented formally in a **Software Requirement Specification (SRS)**.
    - Example: *"The system shall allow customers to place online food orders and track the status in real time."*
4. **Requirement Validation**
    - Conduct reviews and walkthroughs with stakeholders.
    - Ensure requirements are **complete, consistent, feasible, and verifiable**.
    - Example: Confirm with restaurant management whether loyalty rewards should be added now or later.
5. **Requirement Management**
    - Maintain a requirements baseline and track changes.
    - Example: If management later requests "AI-based recommendation of dishes," it can be added as a change request.

Appraisal to the Management:

- By applying the **Requirement Engineering Process**, we have systematically gathered the needs of the restaurant.
- The project should initially focus on **core functional requirements**: online menu, ordering, payment, and delivery tracking.
- Gradually, **advanced features** like AI-driven recommendations, chatbot-based ordering, and integration with food delivery platforms (Zomato, Swiggy) can be added.

- Following RE ensures that the final software will align with **business goals, customer satisfaction, and smooth operations**.

Conclusion:

The Requirement Engineering Process provides a structured way to capture and validate the needs of the restaurant. By focusing on both functional and non-functional requirements, and validating them with stakeholders, the software organization can ensure a **robust, scalable, and user-friendly online restaurant management system**.

Q2)B)1) A food restaurant management have decided to go online for 05 performing all of their activities. The food restaurant management has given the project to your software organization to develop the project, You being the employee of the software organization has been tasked by the management to represent the flow of the data in the software. Analyze the scenario and summarize in a data flow diagram at Level 0 and Level 1.

The food restaurant wants to move online for its operations such as order placement, payment, and delivery. To represent the flow of data in the software, a **Data Flow Diagram (DFD)** is prepared.

Level 0 DFD (Context Diagram)

This shows the system as a **single process** with its main interactions.

Entities:

- **Customer** – places order, makes payment, receives confirmation.
- **Restaurant Admin** – manages menu, confirms orders, updates status.
- **Delivery Staff** – gets assigned delivery details.
- **Payment Gateway** – processes payments.

Process (System): Online Restaurant Management System

Data Flows:

- Customer → (Order, Payment Info) → System
- System → (Order Confirmation, Delivery Status) → Customer

- Admin → (Menu Updates, Order Updates) → System
- System → (Order Requests, Delivery Details) → Delivery Staff
- System ↔ Payment Gateway (Payment Info / Payment Confirmation)

Text Representation of Level 0 DFD:

```
[Customer] → (Order, Payment) → [Online Restaurant System] → (Order Confirmation, Delivery Status) → [Customer]

[Admin] → (Menu Data, Order Updates) → [Online Restaurant System]

[Online Restaurant System] → (Delivery Assignment) → [Delivery Staff]

[Online Restaurant System] ↔ (Payment Info, Payment Confirmation) ↔ [Payment Gateway]
```

Level 1 DFD (Detailed View)

Now, the main system is broken into **sub-processes**.

Sub-processes inside the Restaurant System:

1. **1.0 Order Management** – receives orders, checks menu availability.
2. **2.0 Payment Processing** – sends details to Payment Gateway and validates.
3. **3.0 Restaurant Operations** – forwards confirmed orders to kitchen/admin.
4. **4.0 Delivery Management** – assigns delivery and tracks status.
5. **5.0 Menu & User Management** – manages menu items, discounts, and customer details.

Data Stores:

- **D1: Customer Database** (customer accounts, order history)
- **D2: Menu Database** (menu items, prices)
- **D3: Order Database** (current/past orders)

Data Flows:

- Customer → Order Management → (Order Details stored in Order DB)
- Order Management → Payment Processing → Payment Gateway → Confirmation → Back to Order Management
- Order Management → Restaurant Operations → Kitchen/Admin
- Restaurant Operations → Delivery Management → Delivery Staff
- Admin → Menu & User Management → Menu Database (updates)
- Customer ↔ Menu & User Management (menu browsing, account login)

Text Representation of Level 1 DFD:

```
[Customer] → (Order Details) → (1.0 Order Management) → [D3: Order DB]

(1.0 Order Management) → (2.0 Payment Processing) → [Payment Gateway] ↔ (Payment Confirmation)

(1.0 Order Management) → (3.0 Restaurant Operations) → [Admin/Kitchen]

(3.0 Restaurant Operations) → (4.0 Delivery Management) → [Delivery Staff]

[Admin] → (Menu Updates) → (5.0 Menu & User Management) → [D2: Menu DB]

[Customer] ↔ (5.0 Menu & User Management) ↔ [D1: Customer DB]
```

Summary to Management:

- **Level 0 DFD** shows the overall interaction of customers, admin, payment gateway, and delivery staff with the online restaurant system.

- **Level 1 DFD** breaks down the system into sub-processes like Order Management, Payment Processing, Restaurant Operations, Delivery Management, and Menu Management, with supporting databases.

- This structured flow ensures that data moves efficiently across customers, staff, and management, providing a smooth online restaurant experience.

Q2)B)2) A food restaurant management have decided to go online for performing all of their activities. The food restaurant management has given the project to your software organization to develop the project, You being the employee of the software organization has been tasked by the management to prepare the software requirement specification. Analyze the scenario and summarize the requirements in software requirement specification document.

Software Requirement Specification (SRS) Document

Project Title: Online Restaurant Management System

Prepared by: [Your Software Organization]
Date: [Insert Date]

1. Introduction

1.1 Purpose

The purpose of this SRS is to define the requirements for an **Online Restaurant Management System (ORMS)** that allows the restaurant to manage online ordering, reservations, payments, and delivery tracking.

1.2 Scope

The system will allow customers to:

- Browse menu items.

- Place online food orders and make reservations.

- Make online payments or choose cash on delivery.

- Track order and delivery status.

The restaurant staff and admin will:

- Manage menu, pricing, and offers.
- Handle incoming orders and reservations.
- Assign delivery staff and track deliveries.
- View reports of sales and customer feedback.

1.3 Definitions, Acronyms, and Abbreviations

- ORMS: Online Restaurant Management System
- SRS: Software Requirement Specification
- Admin: Restaurant Manager/Owner
- DB: Database

---

2. Overall Description

2.1 Product Perspective

The system will be a **web-based and mobile-enabled platform** that integrates ordering, payment, and delivery management. It will interact with:

- **Customers** (front-end users)
- **Admins/Restaurant staff** (back-end users)
- **Delivery staff**
- **Payment Gateway** (third-party service)

2.2 Product Functions

- Customer account creation and login.
- Menu browsing and order placement.
- Table reservation system.
- Payment integration (UPI, credit/debit card, wallets, COD).

- Delivery assignment and tracking.
- Admin control panel for managing menu, staff, and reports.

2.3 User Characteristics

- Customers: Non-technical users who want simple ordering.
- Admin/Staff: Semi-technical users who manage orders and inventory.
- Delivery staff: Access only to assigned delivery details.

2.4 Constraints

- System must be available 24/7.
- Must comply with data privacy and payment security regulations.
- Mobile-responsive UI is mandatory.

2.5 Assumptions and Dependencies

- Stable internet connectivity for users.
- Reliable third-party payment gateway.
- GPS integration for delivery tracking.

---

3. Specific Requirements

3.1 Functional Requirements

1. The system shall allow customers to register and log in.
2. The system shall allow customers to browse the menu and add items to the cart.
3. The system shall process online payments securely.
4. The system shall allow customers to book a table.
5. The system shall allow customers to track the status of their order.
6. The system shall notify customers via email/SMS about order updates.

7. The system shall allow admin to add, update, or delete menu items.
8. The system shall allow admin to manage discounts and promotional offers.
9. The system shall allow admin to assign orders to delivery staff.
10. The system shall generate sales and feedback reports.

3.2 Non-Functional Requirements

1. **Performance:** System should support at least 500 concurrent users.
2. **Reliability:** 99% uptime to ensure uninterrupted service.
3. **Security:** Data encryption, secure login, and PCI-DSS compliance for payments.
4. **Usability:** User-friendly interface, accessible on desktop and mobile.
5. **Scalability:** System should be scalable to support multiple branches in the future.

---

4. External Interface Requirements

- **User Interface:** Responsive web application with simple navigation.
- **Hardware Interface:** Compatible with mobile, tablets, and desktop systems.
- **Software Interface:** Payment gateway, SMS/email API, and GPS tracking API.
- **Communication Interface:** HTTPS for secure communication.

---

5. Conclusion

This SRS document outlines the requirements for the Online Restaurant Management System. By implementing the defined functional and non-functional requirements, the software will provide an efficient, secure, and user-friendly platform for both customers and restaurant staff, ensuring smooth online operations.

Q.3 A) A software project manager has decided to make use of the software project metrics to adapt to project workflow and technical activities. Interpret the intent of software project metrics.

Answer:

Software project metrics are **quantitative measures** used by project managers to assess, monitor, and control the progress, quality, productivity, and efficiency of a software project. They provide objective data to guide decision-making and ensure that the project stays on track.

1. **Project Monitoring and Control**
    - Metrics help track project progress against planned schedules, budgets, and resources.
    - Example: *Effort variance* and *schedule variance* metrics indicate whether the project is ahead or behind schedule.
2. **Improving Quality**
    - Metrics such as *defect density*, *mean time to failure*, and *code complexity* help ensure that the software product meets the required quality standards.
3. **Productivity Measurement**
    - By measuring *lines of code (LOC)*, *function points*, or *story points per sprint*, managers can evaluate team productivity and identify bottlenecks.
4. **Risk Management**
    - Metrics highlight potential risks early.
    - Example: A rising defect rate in early testing signals possible quality issues in later stages.
5. **Resource Optimization**
    - Metrics assist in understanding workload distribution and resource utilization, ensuring efficient use of manpower, time, and cost.
6. **Process Improvement**
    - Historical metrics can be analyzed to improve future estimates, processes, and practices.
    - Example: *Cost of Quality (CoQ)* and *Rework Percentage* help refine development methodologies.
7. **Decision-Making Support**

- Metrics provide factual evidence to support managerial decisions, such as whether to add resources, change timelines, or re-prioritize tasks.

Examples of Common Software Project Metrics:

- **Schedule Variance (SV)** – Difference between planned and actual project schedule.
- **Cost Variance (CV)** – Difference between budgeted and actual cost.
- **Defect Density** – Number of defects per KLOC (thousand lines of code).
- **Velocity (Agile metric)** – Number of story points completed per sprint.
- **Customer Satisfaction Index** – Feedback-based metric on delivered product.

Conclusion:

The intent of software project metrics is to provide **visibility, control, and improvement** throughout the software development lifecycle. By using metrics, project managers can adapt workflows, enhance technical activities, minimize risks, and ensure timely delivery of a high-quality software product.

Q3)B)2) Consider the below data flow diagram of the SafeHome software, 05 the function points. Assume that Σ(fi) is 46, a moderately complex product, productivity rate is 15 FP person-month and the labour cost is Rs. 21800 person-month. Calculate the total project estimated cost for given system.

Given

- Sum of general system characteristics: $\sum f_i = 46$
- Complexity: **moderate** → use **VAF** (Value Adjustment Factor)
- Productivity rate: $15\ \text{FP/person-month}$
- Labour cost: ₹ **21,800** per person-month

1) Compute VAF

$$\text{VAF} = 0.65 + 0.01 \times \sum f_i = 0.65 + 0.01 \times 46 = 0.65 + 0.46 = 1.11$$

2) Adjusted Function Points (AFP)

Let **UFP** be the unadjusted function points from the DFD count (from the previous sub-question).

$$\text{AFP} = \text{UFP} \times \text{VAF} = \text{UFP} \times 1.11$$

3) Effort (person-months)

$$\text{Effort} = \frac{\text{AFP}}{\text{Productivity}} = \frac{1.11 \times \text{UFP}}{15} = 0.074 \times \text{UFP person-months}$$

4) Estimated Cost

$$\text{Cost} = \text{Effort} \times \text{Labour cost} = (0.074 \times \text{UFP}) \times 21{,}800 = 1{,}613.2 \times \text{UFP rupees}$$

Final expression (plug in your UFP):

$$\boxed{\text{Total estimated project cost} = 1{,}613.2 \times \text{UFP}\ (₹)}$$

> Example (only if your earlier UFP was, say, 120):
> Cost $= 1{,}613.2 \times 120 = ₹193{,}584.$

So, once you insert the UFP you computed from the SafeHome DFD, multiply by **₹1,613.2** to get the total cost.