CSC584: Assignment 1
Submitted by: Malika Hafiza Pasha

Chapter 1: Introduction

1. **What is the most important difference between generic software product development and custom software development? What might this mean in practice for users of generic software products?**

Answer:

Difference between generic software product development and custom software development:

To provide a solution that works for everyone, generic software product development prioritizes consistency, cost-effectiveness, and a wide audience. However, in order to meet each client's specific demands, custom software development prioritizes flexibility, customization, and exact alignment with client specifications.

Users of generic software products:

- Standard features are offered for shared uses.
- There aren't many customizing options.
- Its affordability is aided by a big user base.
- Anticipate neighborhood support and regular updates.
- Scalability and industry standards are typically incorporated.
- Deep customization is sacrificed, even as onboarding is made simpler.

2. **What are the four important attributes that all professional software should have? Suggest four other attributes that may sometimes be significant and describe each of these attributes.**

Answer:

Four important attributes that all professional software should have:

a. **Reliability**:
Software that is dependable constantly operates as planned in a variety of circumstances. It should not have any serious flaws or defects that could cause crashes, data corruption, or system failures. For mission-critical applications in particular, reliability is essential because it guarantees that users can rely on the program to function as intended.

b. **Efficiency**:
Software that is efficient makes good use of the system resources (CPU, memory, disk space, etc.) and doesn't waste anything. It should respond to user input swiftly and carry out activities efficiently. Optimizing performance and cutting expenses associated with operations require software that is efficient.

c. **Maintainability**:
Updating, extending, and changing software that is maintainable is simple. Because of its clear, well-documented code, developers can make modifications to it without creating new problems. Maintainability requires sufficient documentation, modular architecture, and good coding techniques.

d. **Security**:
Sensitive information is shielded by secure software from harmful assaults, unauthorized access, and weaknesses. To protect user data and system integrity, it should put authentication, authorization, encryption, and other security mechanisms in place.

Four other significant attributes are:

a. **Usability**:
   Software that is easy to use and has an intuitive interface makes it possible for users to do activities without difficulty. Productivity and user happiness are increased by a well-designed user experience.

b. **Scalability**:
   Software that is scalable can manage bigger datasets or higher workloads without experiencing a noticeable decrease in performance. Applications that are anticipated to grow over time, like databases and web services, require scalability.

c. **Interoperability**:
   Even if separate programs or systems were created independently, interoperable software can interact and function with them without any problems. Integration with external tools and services depends on it.

d. **Flexibility**:
   Software that is flexible can change to meet new requirements without requiring significant changes to the code. It permits the inclusion of new features or modifications to accommodate changing user requirements.

3. Based on your own knowledge of some of the application types discussed in section 1.1.2, explain, with examples, why different application types require specialized software engineering techniques to support their design and development. You must identify at least three reasons with examples.

Answer:

Different application types require specialized software engineering techniques to support their design and development due to their unique requirements, constraints, and characteristics. Examples for each application type mentioned:

a. Stand-alone applications:
   These are specialized software programs that operate on a single machine. To guarantee effective resource use and user-friendly interfaces, specialized techniques are required. For instance, to improve usability and efficiency, word processors such as Microsoft Word require strategies for text processing, formatting, and user interface design.

b. Interactive transaction-based applications:
   Specific methods are needed for applications such as e-commerce platforms and online banking systems to manage many user interactions, maintain data integrity, and facilitate safe transactions. To successfully enable these functionalities, techniques like database administration, transaction processing, and encryption are necessary.

c. Embedded control systems:
   Devices with embedded systems, such as medical or automotive control systems, need specific methods to guarantee real-time responsiveness, dependability, and safety. To achieve strict performance and safety criteria, methods like low-level programming, hardware-software co-design, and real-time operating systems are essential.

d. Batch processing systems:

These systems, which are frequently employed in data processing and analysis, need for methods for batching and processing massive amounts of data effectively and on time. To maximize processing time and resource utilization, strategies including data segmentation, job scheduling, and parallel processing are crucial. For instance, in order to process payroll for a large number of employees, a payroll processing system might need to use batch processing techniques.

e. Entertainment systems:

Specialized methods are needed for complicated graphics rendering, audio processing, and user interactions in video games and multimedia applications. To produce immersive and captivating user experiences, techniques including sound processing, 3D rendering algorithms, and gaming engine development are required.

f. Systems for modeling and simulation:

Specialized methodologies are needed for applications used in engineering or science modeling in order to accurately represent complicated phenomena and reproduce real-world settings. To guarantee accuracy and efficiency in simulations, strategies including high-performance computing, algorithm optimization, and numerical approaches are crucial.

g. Data collection systems:

Systems that gather data, such sensor networks or Internet of Things devices, need specific methods to manage data gathering, storing, and analyzing. It is essential to employ strategies like data compression, streaming processing, and distributed computing in order to effectively handle and handle the massive amounts of data that these systems produce.

h. Systems of systems:

Smart cities and integrated healthcare systems are examples of complex systems made up of interconnected subsystems that require specific strategies to guarantee interoperability, scalability, and durability. It takes strategies like middleware development, system integration, and service-oriented architecture to efficiently plan, organize, and control interactions across various subsystems.

Chapter 2: Software processes

4. Consider the reuse-based process model shown in Figure 2.3. Explain why it is essential to have two separate requirements engineering activities in the process.

Answer:

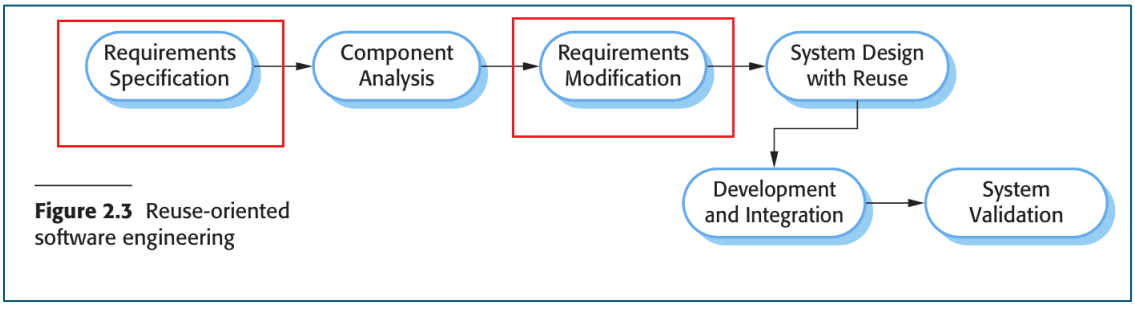

Figure 2.3 Reuse-oriented software engineering

In the reuse-based process model depicted in Figure 2.3, having two separate requirements engineering activities, namely "**Requirement Specification**" and "**Requirement Modification**," is crucial for the following reasons:

a. **Clarity in Requirements Identification:**

The "**Requirement Specification**" task focuses on gathering, examining, and recording the needs particular to the newly designed system. This stage guarantees that the needs, goals, and functional requirements of the stakeholders for the new system are well understood. By dividing out this task, the project team will be able to focus entirely on specifying the special needs of the new system, free from the impact of pre-existing parts or solutions.

b. **Adaptation of Reused Components:**

The purpose of the "**Requirement Modification**" activity is to determine and take care of any adjustments or changes needed so that the repurposed parts work with the new system. Reused parts might not match the new system's specifications exactly because of functional, interface, or non-functional differences. To easily integrate the reused components into the new system, this step entails examining inconsistencies between the needs of the new system and the capabilities of the reused components, as well as establishing change requirements.

c. **Risk Mitigation:**

Requirement modification and specification should be kept apart to reduce the hazards involved in component reuse. Any gaps or incompatibilities with repurposed components can be found early in the process by outlining the requirements of the new system precisely from the beginning. This reduces the possibility of project delays or failures by enabling proactive steps to be made to address these concerns during the requirement modification process.

d. **Efficient Resource Utilization:**

The project team may more efficiently allocate resources based on the distinct goals of each phase when there are distinct activities. While requirement modification may entail technical assessments and modifications to existing components, requirement specification may necessitate considerable stakeholder participation and analysis. Resources can be allocated correctly to guarantee that both aspects are fully covered by clearly defining these activities.

5. **Explain why change is inevitable in complex systems and give examples (apart from prototyping and incremental delivery) of software process activities that help predict changes and make the software being developed more resilient to change.**

Answer:

Complex systems are always going to change for a variety of reasons, including changing needs, new technology, and outside influences. Software development initiatives may be significantly impacted by these modifications. In addition to incremental delivery and prototyping, there are several software process activities that aid in change prediction and increase the adaptability of the product under development.

Developers can better anticipate and accommodate changes in complex systems by adding these software process activities, which strengthens the program's resilience and adaptability to changing needs and outside influences. They are:

a. **Requirements Elicitation and Analysis:**

Potential modifications can be foreseen and taken into consideration during the software development process by carefully examining and interpreting the requirements of the product. Potential areas for improvement can be found with the aid of techniques like user story mapping and use case modeling.

b. **Risk Management:**
Changes can be lessened in impact by recognizing and controlling risks related to the software development process. Performing risk analyses and putting risk reduction plans into action can help the software become more flexible in the future.

c. **Agile Development Practices:**
Software development is made more flexible and adaptable by agile approaches like Scrum and Kanban. Changes may be smoothly implemented with frequent iterations, ongoing feedback, and constant communication with stakeholders.

d. **Modularity and Component-Based Development:**
Modular software design facilitates simpler individual component replacement and customization. Because of this modular design, developers can adapt to changes by changing or replacing individual parts of the system without impacting the system.

e. **Version Control and Configuration Management:**
It is easier to track software modifications and make new changes easier to integrate or roll back when version control systems and configuration management tools are used. This guarantees that the software will not break quickly and can be readily modified as needed.

6. Discuss and describe the four main activities of a software engineering process.

Answer:

a. **Requirements Analysis and Specification:**

Stakeholder requirements are gathered and examined in the first step to make sure they are precise and comprehensive. To comprehend user needs, methods like prototyping and interviews are employed. A precise requirements specification document that directs the development process is the result.

b. **Design:**

Architects and designers draft a blueprint for the software solution throughout the design process, outlining its parts, structure, and data flows. Design choices guarantee that the program satisfies specifications and is both scalable and maintainable. Depending on the project, several design approaches such as object-oriented design are employed. A thorough design document that directs implementation is the result.

c. **Implementation:**

During implementation, developers write code based on design specifications and coding standards. They integrate third-party components and conduct unit testing to ensure module functionality. Collaboration and version control tools facilitate teamwork. The output is an executable software product ready for testing and evaluation.

d. **Testing and Validation:**

The software is tested after it is implemented to make sure it satisfies the requirements. To ensure functionality, performance, and dependability, testing consists of unit, integration, system, and acceptance testing. Test cases

mimic different situations in order to find flaws. Debugging is done using feedback, and before deployment, there may be several iterations. A verified software product that satisfies quality requirements is the result.

Chapter 3: Agile Software Development

7. Explain how the principles underlying agile methods lead to the accelerated development and deployment of software.

Answer:

Agile methods are a set of concepts and practices for software development that prioritize customer demands, flexibility, and collaboration. These concepts facilitate the faster development and deployment of software by promoting a more adaptable and efficient development process. The core principles of Agile accelerate software development and release in the following ways.

a. **Customer-Centric Approach**:
Agile methodology prioritizes frequent feedback from customers and involves them in the development process, which promotes customer satisfaction. By taking this strategy, the risk of creating the incorrect product is reduced and the software is guaranteed to keep up with changing client wants and preferences. Agile makes it possible to offer worthwhile features more quickly by keeping a customer-centric emphasis.

b. **Iterative and Incremental Development**:
Agile breaks down projects into manageable, two- to four-week-long cycles, during which developers prioritize and deliver features. This method of working small promotes ongoing development and makes it possible for interested parties to see results fast.

c. **Collaborative Teams**:
Agile involves close collaboration between developers, testers, designers, and business representatives in cross-functional teams. This cooperative method facilitates quicker decision-making and problem-solving while guaranteeing that all team members agree with the project's objectives.

d. **Adaptive Planning**:
Agile recognizes and welcomes the notion that requirements may change over time. Short planning cycles allow for flexibility and the ability to make changes in response to changing needs or market conditions.

e. **Frequent Inspections and Adaptation**:
Frequent gatherings, such as Agile sprint reviews and daily stand-ups, help identify problems early and correct course. Teams avoid delays brought on by inflexible project plans by quickly adjusting to obstacles and feedback.

f. **Empowered Teams**:
Agile teams have the autonomy to decide how best to accomplish their objectives and are self-organizing. Team members are motivated by this autonomy, which also expedites decision-making.

g. **Continuous Integration and Testing**:
Agile encourages frequent integration of code changes into the main codebase, hence promoting continuous integration. By preventing errors from being introduced by new code, automated testing reduces the amount of time needed to fix bugs.

h. **Working Software as a Measure of Progress**:
Agile places a higher priority on the delivery of functional software at the conclusion of each iteration than it does on extensive documentation. This methodology facilitates expedited validation of advancements and guarantees the consistent production of concrete outcomes.

i. **Transparent Communication**:
Open and honest communication between team members and stakeholders is encouraged by agile. By identifying and resolving issues early on, this transparency helps to avoid bottlenecks.

j. **Customer Acceptance Testing**:
Agile processes verify client acceptability of regularly supplied increments to make sure they meet customer expectations and allow for timely deployment when they're ready.

k. **Continuous Delivery and Deployment**:
Continuous Integration and Continuous Delivery (CI/CD) is a common technique used by agile teams to automate and optimize the deployment process. The time and effort required to move software from development to production environments is greatly decreased by this automation.

l. **Risk Mitigation**:
Early risk assessment and mitigation are given top priority by agile principles. Agile ensures easier project execution by reducing the possibility of significant disruptions through ongoing monitoring and modification.

8. Compare and contrast the Scrum approach to project management with conventional plan-based approaches. Your comparison should be based on the effectiveness of each approach for planning the allocation of people to projects, estimating the cost of projects, maintaining team cohesion, and managing changes in project team membership.

Answer:

The Scrum approach to project management and conventional plan-based approaches differ significantly in their methods and effectiveness for various project aspects.

Parameters	Scrum	Conventional
Effectiveness applicability	Ideal for tasks of a small to medium size.	Ideal for large-scale projects requiring security and safety.
Allocation of People to Projects	Scrum encourages self-organizing teams, giving members the freedom to choose how best to divide their knowledge and abilities. When the team must make decisions, the Scrum Master helps.	Plan-based techniques usually entail project managers allocating resources in a top-down manner according to predetermined roles and responsibilities.
Estimating the Cost of Projects	Scrum emphasizes incremental value delivery, enabling early feedback and modification. The team's velocity, or the quantity of work they can complete in a specific amount of time, is frequently used to estimate costs.	Plan-based techniques frequently entail thorough upfront cost estimation based on resource allocation and task division.

Maintaining Team Cohesion	Scrum promotes excellent team cohesion by emphasizing transparency and collaboration.	Plan-based approaches could give less weight to team cohesion and dynamics because each team member is primarily concerned with completing their allocated responsibilities.
Managing Changes in Project Team Membership	Scrum teams can more easily adjust to changes in team membership since they are self-organizing. With the current team structure in place, new people can easily fit in and contribute.	Since roles and duties in plan-based approaches are generally established and necessitate reassignment, managing changes in team membership may require extra administrative work.

9. Why is it necessary to introduce some methods and documentation from plan-based approaches when scaling agile methods to larger projects that are developed by distributed development teams.

Answer:

When scaling Agile methods to larger projects built by distributed teams, it is vital to introduce techniques and documentation from plan-based approaches for a number of reasons:

a. Plan-based approaches provide comprehensive documentation and methods that are essential for elucidating project goals, specifications, and dependencies. This guarantees alignment and synchronization across scattered teams and larger projects.

b. Agile teams find it necessary to foresee and effectively manage risks as project complexity and team distribution expand, and they can do so by using techniques from plan-based approaches.

c. To guarantee adherence to standards, regulatory compliance requirements in industries such as banking, healthcare, and government may call for documentation and procedures from plan-based approaches.

d. Plan-based documentation's thorough reporting and communication methods are essential for managing expectations and keeping stakeholders informed, particularly in distant situations.

e. To meet the scalability needs of larger projects, plan-based approaches may need to be added to agile methodologies in addition to extra planning, documentation, and coordination systems.

f. To ensure seamless integration and collaboration, plan-based methods offer tools and procedures for managing dependencies between teams, components, and external stakeholders in larger projects.

g. Integrating Agile methods with plan-based approaches' quality assurance procedures and documentation standards guarantees that quality is a top concern in projects that are bigger, more intricate, and created by remote teams.