Chapter 2:

2.1 Suggest the most appropriate generic software process model that might be used as a basis for managing the development of the following systems:

- A system to control antilock braking in a car
  - Safety Critical System
  - Plan Driven approach with requirements analysed carefully
  - Waterfall model most appropriate with formal transformations between different development stages.
- A virtual reality system to support software maintenance
  - Cutting edge and UI dependent for usability
  - Incremental or Spiral with some UI prototyping
  - Agile process may be used
- A university accounting system that replaces an existing system
  - Requirements fairly well known
  - Reuse based approach is appropriate
- An interactive travel planning system that helps users plan journeys with the lowest environmental impact
  - Complex UI which is stable and reliable
  - Incremental development approach most appropriate
  - System requirements will change as user experience is gained

2.2 Incremental software development could be very effectively used for customers who did not have a clear idea about the systems needed for their operations. Discuss

- Based on the idea of developing initial implementation, getting user feedback and evolving software through several versions.
- Either plan driven, agile or mixture.
- Customer can evaluate systems at a relatively early stage.

2.3 Consider the integration and configuration process model shown in Figure 2.3. Explain why it is essential to repeat the requirements engineering activity in the process.

- Firstly requirements spec involve proposing initial requirements for the system
  - Brief descriptions
- Refinement stage, reusable components and applications discovered for refinement.
  - Modified to reflect available components and system spec is redefined.

2.4 Suggest why it is important to make a distinction between developing the user requirements and developing system requirements in the requirements engineering process

1) User Requirements describe the system functions and features from the perspective of a user. These are usually abstract. System requirements provide a more detailed explanation of the procedure.

2) User requirements are written in plain and natural language. System requirements are more detailed with specific specifications that could be part of a contract.

3) User requirements start with gathering information, identifying important aspects, and understanding them. System Requirements can only be implemented after the user requirements are understood and finalised.
2.5 Using an example, explain why the design activities of architectural design, database design, interface design and component design are interdependent

Design of architecture:
- System overall structure is defined.
- Discuss reason for interdependency

Design of Database:
- Define structure of database as representation.

Design of interface:
- Interface defined here.
- Ease of use

Component design and selection
- Reusable components matching requirements are defined here

All design activities above should be followed and integrated. Therefore these work together and can be called interdependent

2.6 Explain why software testing should always be incremental, staged activity. Are programmers the best people to test the programs that they have developed?

Software testing:
- Each program is tested for correct functionality
- Complete program divided into small modules, tested individually. (UNIT TESTING)
- Then whole system is tested with the programs tested above. (SYSTEM TESTING)
- Beta version released and end user checks (CUSTOMER TESTING)

Programmers are not the best persons
- Hard to find own errors.
- Best skills to develop but not to test.

2.7 Imagine that a government wants a software program that helps to keep track of the utilization of the country’s vast mineral resources. Although the requirements put forward by the government were not very clear, a software company was tasked with the development of a prototype. The government found the prototype impressive, and asked it be extended to be the actual system that would be used. Discuss the pros and cons of taking this approach

Pros:
- Allows changes to prototype
- Customer sees prototype and modifications made before design is created
- Better implementation
- Satisfied customer

Cons:
- Additional cost for prototype creation
- Delays due to extra time creating prototypes
2.8 You have developed a prototype of a software system and your manager is very impressed by it. She proposes that it should be put into use as a production system, with new features added as required. This avoids the expense of system development and makes the systems immediately useful. Write a short report for your manager explaining why prototype systems should not normally be used as production systems

- Prototype anticipates changes required
- Requirements engineering prototype helps with elicitation and validation
- System design process: used to explore software solutions in UI development
- Minimal UI and not intuitive
- No error detection
- Vague error messages
- Not viewed as high quality product, only development aids.

2.9 Suggest two advantages and two disadvantages of the approach to process assessment and improvement that is embodied in the SEI’s Capability Maturity framework

Adv:
- Focused on software engineering processes and practices used
- Led to significant improvements in capabilities

Disadv:
- Too much overhead in formal process improvement in small companies
- Maturity estimation with agile processes is difficult

2.10 Historically, the introduction of technology has caused profound changes in the labour market and, temporarily at least, displaced people from jobs. Discuss whether the introduction of extensive process automation is likely to have the same consequences for software engineers. If you don’t think it will, explain why not. If you think that it will reduce job opportunities, is it ethical for the engineers affected to passively or actively resist the introduction of this technology?

- Reduce human error in code creation
- Potential to produce similar or better software than conventionally produced software
- Cost reduction

- Standardised components used, increasing software reliability and cost reduction in future maintenance
- Automation assists software to address primary issues in the development process. (complexity, reliability, and productivity)
3.1 At the end of their study program, students in a software engineering course are typically expected to complete a major project. Explain how the agile methodology may be very useful for the students to use in this case

- Team centered, assisting with interaction.
- Phase release, showing updates as they work
- Specification, design, and implementation are interleaved.
- No detailed system spec and design docs is minimized.

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

- Individual and interactions over processes and tools
  - This means that the team can focus on the development of working software
- Working software over comprehensive documentation
  - Rather the programmer’s time is focussed on the development and testing of code.
- Customer collaboration over contract negotiation
  - This allows useful functionality to be developed and delivered earlier than would be possible if contracts were required.
- Responding to change over following a plan
  - There is significant overhead in changing plans to accommodate change and the inflexibility of a plan means that work may be done that is later discarded.

3.3 Extreme programming expresses user requirements as stories, with each story written on a card. Discuss the advantages and disadvantages of this approach to requirements description

Adv of stories:
- Represent real situations arising so systems support common user operations
- Easy for users to understand and critique stories
- Represent increments of functionality

Disadv:
- Liable to be incomplete
- Focus on functional requirements
- Representing system requirements such as performance and reliability are impossible
- Relationship between architecture and user stories is unclear.

3.4 In test-first development, tests are written before the code. Explain how the test suite may compromise the quality of the software system being developed.

- The test suite could become outdated by the time the actual code is written and not test every vulnerability that the code may have.
- If tests aren’t reviewed, and further tests written, undetected bugs may be in system release.
3.5 Suggest four reasons why the productivity rate of programmers working as a pair might be more than half that of two programmers working individually

- Pair programming leads to continuous informal reviewing. Discovers bug quicker than individual testing.
- Informal sharing is implicit. Reducing need for documentation and time required for another’s work load.
- Encourages refactoring – reducing costs of subsequent development and changes in the future can be done quicker.
- People are likely to spend less time in fine-grain optimization. Both focus on essential features which are produced quicker.

3.6 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.

<table>
<thead>
<tr>
<th>Effectiveness applicability:</th>
<th>Scrum Approach:</th>
<th>Conventional plan-based Approaches:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning allocation of people to projects:</td>
<td>The whole team is involved in the project. No project managers. Planning is participated by everyone.</td>
<td>A manager will break down the work parts and assigns the parts to teams of members involved.</td>
</tr>
<tr>
<td>Estimating Cost on projects:</td>
<td>Easier to estimate costs as the project develops throughout the increments of software development, as the amount of time needed for development can be predicted easily.</td>
<td>Determined during initial planning phases. Cost is difficult to determine due to the amount of time needed and product feasibility being undeterminable.</td>
</tr>
<tr>
<td>Maintaining team cohesion:</td>
<td>Teams are always working together on the same increment and everything is visible to everyone. Frequent meetings held and tracks backlog of work</td>
<td>Teams are working on different parts of a system, therefore smaller teams work together as opposed to one large team working together, not knowing what other teams are doing</td>
</tr>
<tr>
<td>Managing changes in project team membership:</td>
<td>Lesser number of members in the project team due to all everybody working together.</td>
<td>Requires larger amounts of members to ensure the teams have sufficient members.</td>
</tr>
</tbody>
</table>

3.7 To reduce costs and the environmental impact of commuting, your company decides to close a number of offices and to provide support for staff to work from home. However, the senior management who introduce the policy are unaware that software is developed using Scrum. Explain how you could use technology to support Scrum in a distributed environment to make this possible. What problems are you likely to encounter using this approach?

Use of technology:
- Video conferencing
- Scrum master should be located with development team to be aware of everyday problems
- Product owner should visit developers to establish good relationships with them
- Real time communication through informal communication, i.e. instant message and video calls.
- Continuous integration, all team members can be aware of state of product at any time
- Common development environment for all teams

Problems:
- Development requirement daily meetings but not possible in distributed environment
• Communication gap can occur between members
• Changes lead to slow the entire development of project
• Pair programming benefits detection and evaluation of errors.

3.8 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?

1. Project Planning:
   a. Essential when developing software with larger teams to ensure right people are available when needed
   b. Ensure delivery schedules are aligned

2. Requirements analysis
   a. Important to decide how to distribute work across teams and ensure each team has some understanding of other teams’ work

3. Design documentation:
   a. Important so teams develop independently with having access to software under development.

4. Risk management:
   a. Required to ensure all teams understand risks faced and organize their work to minimize these risks.
   b. Useful to cope with different delivery schedules.

3.9 Explain why agile methods may not work well in organizations that have teams with a wide range of skills and abilities and well-established processes.

Having to switch from their well-established process to a new process might set them back on time that they could use to develop.

3.10 One of the problems of having a user closely involved with a software development team is that they “go native.” That is, they adopt the outlook of the development team and lose sight of the needs of their user colleagues. Suggest three ways how you might avoid this problem, and discuss the advantages and disadvantages of each approach.

1. Involve multiple users in the development team.
   a. Different perspectives
   b. Better coverage of user tasks
   c. Disadvantages are cost, and getting user engagement and conflicts.

2. Change the user who is involved with the team.
   a. Multiple perspectives.
   b. Disadv. Each user takes time to be productive

3. Validate user suggestions with other user representatives
   a. Advantages are independent check on suggestions
   b. Disadv. Is slowing down development process to do checks.