1 Introduction

1.2 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?

The essential difference is that in generic software product development, the specification is owned by the product developer. For custom product development, the specification is owned and controlled by the customer. The implications of this are significant – the developer can quickly decide to change the specification in response to some external change (e.g. a competing product) but, when the customer owns the specification, changes have to be negotiated between the customer and the developer and may have contractual implications.

For users of generic products, this means they have no control over the software specification so cannot control the evolution of the product. The developer may decide to include/exclude features and change the user interface. This could have implications for the user's business processes and add extra training costs when new versions of the system are installed. It also may limit the customer's flexibility to change their own business processes.

1.3 What are the four important attributes that all professional software should have? Suggest four other attributes that may sometimes be significant.

Four important attributes are maintainability, dependability, performance and usability. Other attributes that may be significant could be reusability (can it be reused in other applications), distributability (can it be distributed over a network of processors), portability (can it operate on multiple platforms e.g laptop and mobile platforms) and inter-operability (can it work with a wide range of other software systems).

Decompositions of the 4 key attributes e.g. dependability decomposes to security, safety, availability, etc. is also a valid answer to this question.

1.4 Apart from the challenges of heterogeneity, business and social change and trust and security, identify other problems and challenges that software engineering is likely to face in the 21st century (hint: think about the environment).

Problems and challenges for software engineering

There are many possible challenges that could be identified. These include:

1. Developing systems that are energy-efficient. This makes them more usable on low power mobile devices and helps reduce the overall carbon footprint of IT equipment.
2. Developing validation techniques for simulation systems (which will be essential in predicting the extent and planning for climate change).
3. Developing systems for multicultural use
4. Developing systems that can be adapted quickly to new business needs
5. Designing systems for outsourced development
6. Developing systems that are resistant to attack
7. Developing systems that can be adapted and configured by end-users
8. Finding ways of testing, validating and maintaining end-user developed systems

1.5 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.

Different application types require the use of different development techniques for a number of reasons:

1. Costs and frequency of change. Some systems (such as embedded systems in consumer devices) are extremely expensive to change; others, must change frequently in response to changing requirements (e.g. business systems). Systems which are very expensive to change need extensive up-front analysis to ensure that the requirements are consistent and extensive validation to ensure that the system meets its specification. This is not cost-effective for systems that change very rapidly.
2. The most important 'non-functional' requirements. Different systems have different priorities for non-functional requirements. For example, a real-time

control system in an aircraft has safety as its principal priority; an interactive game has responsiveness and usability as its priority. The techniques used to achieve safety are not required for interactive gaming; the extensive UI design required for games is not needed in safety-critical control systems.

3. The software lifetime and delivery schedule. Some software systems have a relatively short lifetime (many web-based systems), others have a lifetime of tens of years (large command and control systems). Some systems have to be delivered quickly if they are to be useful. The techniques used to develop short-lifetime, rapid delivery systems (e.g. use of scripting languages, prototyping, etc.) are inappropriate for long-lifetime systems which require techniques that allow for long-term support such as design modelling.

1.8 Discuss whether professional engineers should be certified in the same way as doctors or lawyers.

These are possible discussion points - any discussion on this will tend to be wide ranging and touch on other issues such as the nature of professionalism, etc.

Advantages of certification

- Certification is a signal to employers of some minimum level of competence.
- Certification improves the public image of the profession.
- Certification generally means establishing and checking educational standards and is therefore a mechanism for ensuring course quality.
- Certification implies responsibility in the event of disputes. Certifying body is likely to be accepted at a national and international level as 'speaking for the profession'.
- Certification may increase the status of software engineers and attract particularly able people into the profession.

Disadvantages of certification

- Certification tends to lead to protectionism where certified members tend not to protect others from criticism.
- Certification does not guarantee competence merely that a minimum standard was reached at the time of certification.
- Certification is expensive and will increase costs to individuals and organisations.
- Certification tends to stultify change. This is a particular problem in an area where technology developments are very rapid.

2 Software Processes

2.1 Giving reasons for your answer based on the type of system being developed, 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 anti-lock braking in a car
- A virtual reality system to support software maintenance
- A university accounting system that replaces an existing system
- An interactive travel planning system that helps users plan journeys with the lowest environmental impact

1. *Anti-lock braking system* This is a safety-critical system so requires a lot of up-front analysis before implementation. It certainly needs a plan-driven approach to development with the requirements carefully analysed. A waterfall model is therefore the most appropriate approach to use, perhaps with formal transformations between the different development stages.
2. *Virtual reality system* This is a system where the requirements will change and there will be an extensive user interface components. Incremental development with, perhaps, some UI prototyping is the most appropriate model. An agile process may be used.
3. *University accounting system* This is a system whose requirements are fairly well-known and which will be used in an environment in conjunction with lots of other systems such as a research grant management system. Therefore, a reuse-based approach is likely to be appropriate for this.
4. *Interactive travel planning system* System with a complex user interface but which must be stable and reliable. An incremental development approach is the most appropriate as the system requirements will change as real user experience with the system is gained.

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.

You need to repeat the requirements engineering activity because it is essential to adapt the system requirements according to the capabilities of the system/components to be reused. These activities are:

1. An initial activity where you understand the function of the system and set out broad requirements for what the system should do. These should be expressed in sufficient detail that you can use them as a basis for deciding of a system/component satisfies some of the requirements and so can be reused.
2. Once systems/components have been selected, you need a more detailed requirements engineering activity to check that the features of the reused software meet the business needs and to identify changes and additions that are required.

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.

There is a fundamental difference between the user and the system requirements that mean they should be considered separately.

1. The user requirements are intended to describe the system's functions and features from a user perspective and it is essential that users understand these requirements. They should be expressed in natural language and may not be expressed in great detail, to allow some implementation flexibility. The people involved in the process must be able to understand the user's environment and application domain.
2. The system requirements are much more detailed than the user requirements and are intended to be a precise specification of the system that may be part of a system contract. They may also be used in situations where development is outsourced and the development team need a complete specification of what should be developed. The system requirements are developed after user requirements have been established.

2.6 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.

Systems must change because as they are installed in an environment the environment adapts to them and this adaptation naturally generates new/different

system requirements. Furthermore, the system's environment is dynamic and constantly generates new requirements as a consequence of changes to the business, business goals and business policies. Unless the system is adapted to reflect these requirements, its facilities will become out-of-step with the facilities needed to support the business and, hence, it will become less useful.

Examples of process activities that support change are:

1. Recording of requirements rationale so that the reason why a requirement is included is known. This helps with future change.
2. Requirements traceability that shows dependencies between requirements and between the requirements and the design/code of the system.
3. Design modeling where the design model documents the structure of the software.
4. Code refactoring that improves code quality and so makes it more amenable to change.

2.9 Suggest two advantages and two disadvantages of the approach to process maturity that is embodied in the SEI's Capability Maturity Framework.

Advantages of process improvement frameworks

1. The approach provides a means of measuring the state of a process and a structured approach to introducing process improvements.
2. It is useful as a way of building on the experience of others in process improvement.

Disadvantages of process improvement frameworks

1. Like any measurement system, there is a tendency to introduce improvements to improve the measured rating rather than concentrate on improvements that meet real business goals.
2. The maturity model approach is expensive and bureaucratic to operate. It is not really suitable for organisations that use agile development.