1. a. **Generic products** are produced to be sold on the open market to any customer who is able to buy them, while **customized products** are commissioned by a particular customer.
   b. More and more systems are now being built with a **generic product** as a base, which is then **adapted to suit the requirements of a particular customer**.

2. It is usually cheaper, in the long run, to use the systematic and organized approach. For most types of systems, the majority of costs are associated with changing the software after it has gone into use.

3. Since there are no absolutes when it comes to safety, he feels an engineer in this situation must make up his own mind about what to do, taking into account the potential for damage, the extent of the damage, and the people affected by the damage.

4. Decrease diversity in software processes across the organization through process standardization.

5. **Cleanroom software engineering**: An example of a formal development process, originally developed by IBM. Software correctness is demonstrated using a formal approach; there is no unit testing for defects in the process.
   
   **Incremental Development**: May not be compatible with large organizations where bureaucratic procedures have evolved over time, for example, to ensure that software properly implements external regulations.
   
   **Waterfall**: Because producing and approving documents can be costly, it is normal to freeze parts of the development, such as specification, and to continue with the later development stages. Problems are left for later resolution, ignored, or programmed around.
   
   **Incremental Development**: Reflects the way that we solve problems – moving toward a solution in a series of steps, backtracking when we realize that we have made a mistake.
   
   **Reuse-based Development**: In addition to inevitable requirements compromises, some control over system evolution is lost as new versions of some components are not under the control of the organization using them.
   
   **Boehm’s Spiral Development**: Combines change avoidance with change tolerance; it assumes that changes are a result of things that can go wrong in a project, and includes explicit management activities to reduce the likelihood and impact of such things.
   
   **RUP**: A phased model that supports iteration in two ways: each phase may be enacted in an iterative way with the results developed incrementally AND the whole set of phases may also be enacted incrementally.
RUP: A practice perspective describes six fundamental best software engineering practices that are recommended for use in systems development.

**Throw-away prototyping:**

![Diagram](image1.png)

**Volere:**

![Diagram](image2.png)

6. **experimental prototyping:** Objective is to evaluate a proposed solution for feasibility / performance.

**“Wizard of OZ” prototyping:** “To realistically evaluate the dynamic behavior of different interface designs for a new chess playing program before implementing its core (chess playing) engine, Company XYZ hired a chess master to simulate the engine in real-time from another city as test subjects played chess with prototypes reflecting the different designs.”

**mockup:** “A non-functioning but realistic looking prototype was developed to test the market appeal of the new devise.”

**vertical prototyping:** “It was decided to develop a prototype that would very realistically reflect only that system functionality that handles emergency operator over-ride situations.”

**simulation:**

![Diagram](image3.png)

**paper prototyping:**

![Diagram](image4.png)

**throw-away prototyping:** May result in:
7. A description of the reason for the requirement.

8. a. Agile methods have to rely on contracts in which the customer pays for the time required for system development rather than the development of a specific set of requirements.
   b. If problems arise then there may be difficult disputes over who is to blame and who should pay for the extra time and resources required to resolve the problems. (I once hired a carpenter to do some out-of-the-ordinary, difficult to pre-specify custom interior work by the hour, and ran into this exact problem.)

9. a. In closed interviews, the stakeholder answers a pre-defined set of questions, while in open interviews, there is no pre-defined agenda.
   b. He points out that there are subtle power relationships between the different people in the organization, and that interviewees may not wish to reveal the actual rather than the theoretical structure to a stranger. In general most people are reluctant to discuss political and (sensitive) organizational issues that may affect the requirements.

10. (1) Sommerville believes that the system requirements document, which tells the software engineer what the system is supposed to do, is critical. (2) Without such knowledge, it is difficult to assess the impact of proposed system changes. (3) Many agile methods collect requirements informally and incrementally and do not create a coherent, formal requirements document.

11. a. A Scrum Master is NOT a project manager. Instead he/she is a facilitator who arranges daily meetings, tracks the backlog of work to be done, records decisions, measures progress against the backlog, and communicates with customers and management outside the team.
   b. Sprints are fixed length (normally 2-4 weeks) and correspond to the development of a release of the system in XP.
   c. The work done is presented to stakeholders, not the Scrum Master.

12. system attribute: Example would be: "system response time under full operational load"

   system goal: Usually suffers from not being verifiable.

   process constraint: Example would be: "Use of the Eclipse IDE is mandatory."

   external requirement: Example would be: "The system is required to meet CENELEC standards EN 50126, EN 50128, and EN 5019 before being approved for deployment."

   domain requirement: Example would be: "At least two fully functioning thrusters are required to reach escape velocity at 14.5038 psi."

   system goal: Example would be: "Should integrate easily with customers’ other systems."

   SRS: Official statement of what is required of system developers; it should include both user and system requirements.

   system constraint: Example would be: "probability of system unavailability must
be ≤ .05"

**interface specification:** Example would be:

```
// Server-side interface specification

func callMe(to string) {
  if to == "hello"
    return "Hello!"
  else
    return "Goodbye!"
}
```

**operational specification:** Example would be:

```
REPEAT
  Select the next (initially, the first) effect.
  Trace back through the graph (right to left).
  First all feasible combinations of connected
  Cause values that result in the effect being
  True.
  For each new set combination found:
    Enter values for each Cause and Effect in a
    new column of the test case coverage matrix.
  UNTIL each effect has been selected.
```

13. **Repository:** Used in data-driven systems where a blackboard model triggers components when particular data becomes available in a shared database.

**Abstract machine model:** Used when there is a requirement for multi-level security.

**Client-server:** Very commonly used run-time organization for distributed systems, connected using Internet protocols.

**Data-flow architecture:** Also known as a "pipe and filter architecture". The name comes from the original Unix system where it was possible to link processes ("filters") using "pipes".

**Information systems:** Application architecture that allows controlled access to a large base of information, such as a library catalog, a flight timetable, or the records of patients in a hospital.

**Data-flow architecture:** Example would be:

![Data-flow Architecture Diagram]

**Abstract machine model:** Example would be:

![Abstract Machine Model Diagram]

14. The isolation ("hiding") of potentially changeable design decisions (possibly in objects, functions, procedures, etc.) to minimize the impact of change.
15. They are used to identify the *other systems in the environment of the system being developed.*

16. **Version management:** Keeping track of the different versions of software components. (Facilities are provided to coordinate development by several programmers; they stop one developer from overwriting code that has been submitted to the system by someone else.)

17. Using GPL open source code in an application requires that the application also be open source.

18. **Beta testing:** End-user testing performed in the customer environment prior to general release.

   **Validation testing:** Undertaken to demonstrate that a system operates as intended.

   **Software inspections:** Involve people examining a system artifact (requirements, design documents, source code, etc.), usually with the aim of discovering anomalies and defects.

   **Regression testing:** Re-running of one or more test cases, after some program change, that ran without revealing faults prior to the change.

   **Software inspections:** V&V activity that can consider quality attributes such as compliance with standards, portability, and maintainability.

   **Partition testing:** Choosing tests from identified groups of inputs that have common attributes and are therefore expected to be processed by a program in the same way.

   **Exhaustive testing:** Would guarantee the program is error-free if no defects were found.

   **Test-driven development:** Involves implementing new program functionality *AFTER* implementing and running a test for that functionality.

   **Defect testing:** Undertaken to expose situations in which the behavior of the software incorrect, undesirable, or does NOT conform to its specification.

   **Operational profile:** Intended to reflect the expected usage of a system in some environment.

   **Alpha testing:** End-user testing performed at the developer’s site

   **Regression testing:** May be automated by combining a keystroke recorder and tool with a data/output comparator

   **Stress testing:** Focus is on typical requirements that systems exhibit “graceful” failures when overloaded.

   **Performance testing:** Usually involves a series of tests where the load on a system is steadily increased.
19. The evolution of systems can rarely be considered in isolation; **evolution in a “systems-rich” environment increases difficulties and costs.** As well as understanding and analyzing the impact of a proposed change on the system itself, **you also have to assess its affect on other systems in the operational environment.** (Note: in civil engineering, Brownfield land means places where new buildings may need to be designed and erected considering the other structures already in place.)

20. a. Program evolution is a self-regulating process. System attributes such as size, time between releases, and the number of reported errors is approximately invariant for each system release.

b. **Structural factors:** Making SMALL program changes/extensions in each release may not satisfy all customer wishes, but reduces the extent of structural degradation that naturally takes place over time and so lessens the risks of causing dependability problems. Making LARGE program changes/extensions may be attractive from a marketing perspective, but can increase the probability of new faults and therefore inhibit further evolution. Hence, the rate by which changes/extensions are made tends to be self-regulating.

**Organizational factors:** Large systems are usually produced by large organizations with internal bureaucracies that set change budgets and control the decision making process. Making decisions concerning the risks, value, and cost of changes is time consuming and often governs (i.e., regulates) the rate of change of the system.

---

**Histogram of Raw Scores**

```
poor  marginal  fair  ok  good  v. good  excellent
```

```
mean (59.5 pts.)
```