1. For other than small, simple software, incremental integration testing strategies are usually preferred to putting all of the components together at once—which is often called “big bang” testing.

2. Thus, all software design and development must take into consideration human-user factors such as how people use software, how people view software, and what humans expect from software.

3. This paradox implies, essentially, that a wicked problem has to be solved once in order to define it clearly and then solved again to create a solution that works.

4. Design goals may include minimization of monetary cost and maximization of reliability, performance, or some other criteria on a wide range of dimensions.

5. Furthermore, writing test cases first forces programmers to think about requirements and design before coding, thus exposing requirements and design problems sooner.

6. One of the most important activities in design is documentation of the design

solution as well as of the tradeoffs for the choices made in the design of the solution.

7. Separating concerns by views allows interested stakeholders to focus on a few things at a time and offers a means of managing complexity.

8. Although some detailed design may be performed prior to construction, much design work is performed during the construction activity.

9. Formal design analysis can be used to detect residual specification and design errors (perhaps caused by imprecision, ambiguity, and sometimes other kinds of mistakes).

10. Often, the impact on quality attributes and tradeoffs among competing quality attributes are the basis for design decisions.

11. A product's life cycle costs are largely influenced by the design of the product.

12. Structured design is generally used

after structured analysis, thus producing (among other things) data flow diagrams and associated process descriptions.

13. Some requirements represent emergent properties of software — that is, requirements that cannot be addressed by a single component but that depend on how all the software components interoperate.

14. Low fidelity prototypes are often preferred to avoid stakeholder "anchoring" on minor, incidental characteristics of a higher quality prototype that can limit design flexibility in unintended ways.

15. Architectural design is the point at which the requirements process overlaps with software or systems design and illustrates how impossible it is to cleanly decouple the two tasks.

16. Flagging potentially volatile requirements can help the software engineer establish a design that is more tolerant of change.

17. Software requirements are often written in natural language, but, in software requirements specification, this may be

supplemented by formal or semiformal descriptions.

18. In practice, therefore, it is almost always impractical to implement the requirements process as a linear, deterministic process in which software requirements are elicited from the stakeholders, baselined, allocated, and handed over to the software development team.

19. A combination of topdown analysis and design methods and bottomup implementation and refactoring methods that meet in the middle could provide the best of both worlds.

20. Primitiveness means the design should be based on patterns that are easy to implement.

21. When designing a user interface, software engineers should be careful to not use more than one metaphor for each concept.

22. Trust management is a design concern; components treated as having a certain degree of trustworthiness should not depend

on less trustworthy components or services.

23. Likewise, it is also helpful if managers of complex projects and programs in which software is a component of the system architecture are aware of the differences that software processes introduce into project management and project measurement.

24. Perhaps the most crucial point in understanding software requirements is that a significant proportion of the requirements will change.

25. In many cases, the software engineer acts as software architect because the process of analyzing and elaborating the requirements demands that the architecture/ design components that will be responsible for satisfying the requirements be identified.