1. **Software**: Programmi informatici con documentazione associata, suddivisi in software generici e personalizzati.
2. **Modello di processo software**: Rappresentazione semplificata di un processo software, con diverse prospettive come flusso di lavoro, flusso di dati, e ruoli/azioni.
3. **Modello a cascata**: Un modello di sviluppo software con fasi distinte e sequenziali: analisi, progettazione, implementazione, test, manutenzione.
4. **Testing black-box**: Un metodo di testing dove la struttura interna del sistema non è conosciuta.
5. **Requisiti non-funzionali**: Vincoli sul sistema come prestazioni, affidabilità e usabilità.
6. **Requisiti utente**: Descrizioni in linguaggio naturale dei servizi e vincoli operativi del sistema.
7. **Extreme Programming**: Un approccio di sviluppo software agile che enfatizza la programmazione in coppia e la consegna incrementale. I test sono scritti prima del codice.
8. **Gestione del progetto**: Include specificazione, progettazione, validazione ed evoluzione del software.
9. **Pianificazione del progetto**: Implica definire obiettivi, attività, tempistiche e risorse.
10. **Stima dei costi del software**: Include metodi per valutare i costi di sviluppo e manutenzione del software.
11. **Tipologia di personale**: Varia in base al progetto e può includere sviluppatori, analisti, tester, ecc.
12. **Giudizio sulla qualità del sistema**: Dipende da fattori come usabilità, affidabilità e performance.
13. **Metodi rapidi di sviluppo software**: Studiati per accelerare la consegna e migliorare la qualità del software.
14. **Requisito**: Una dichiarazione su un servizio o vincolo del sistema.
15. **Utilizzo dei colori**: Linee guida per l'usabilità e l'estetica nelle interfacce utente.
16. **Scrittura di messaggi d’errore**: Devono essere chiari, utili e non tecnici.
17. **Attributi di usabilità**: Include facilità di apprendimento, efficienza e soddisfazione dell'utente.
18. **Funzione dei requisiti**: Definire le funzionalità e i vincoli del sistema.
19. **Requisiti di sistema vs utente**: I requisiti di sistema sono più dettagliati e tecnici.
20. **Facilità d'uso del sistema**: Essenziale per l'accettazione da parte degli utenti.
21. **Caratteristiche dei requisiti**: Chiarezza, completezza, coerenza e verificabilità.
22. **Scrittura dei requisiti utente**: In linguaggio naturale e comprensibile.
23. **Scrittura dei Requisiti di sistema**: Con specifiche tecniche dettagliate.
24. **Modelli organizzativi**: Strutture e processi di un'organizzazione nel contesto del progetto.
25. **Stile di controllo**: Come viene gestita la comunicazione e il flusso di lavoro.
26. **Modelli architettura distribuita**: Include client-server, peer-to-peer, e architettura a strati.
27. **Diagramma UML**: Rappresentazioni grafiche per modellare sistemi software.
28. **Sviluppo agile per software di dimensioni medio-piccole**: Più adatto a progetti agili e flessibili.
29. **COTS (Commercial-off-the-shelf)**: Software pronto all'uso e non personalizzato.
30. **Principi dello sviluppo agile**: Include risposta al cambiamento, iterazione, e collaborazione cliente-sviluppatore.
31. **Scarto di software legacy**: Deciso in base a obsolescenza, costi di mantenimento e adattabilità.
32. **Leggi di Lehman**: Principi che descrivono l'evoluzione dei sistemi software.

33. **Costi nel processo evolutivo**: Include manutenzione, aggiornamento e adattamento.
34. **Misura di produttività**: Valutazione dell'efficienza e dell'efficacia nel processo di sviluppo.
35. **Influenze sulle linee di codice(LOC)**: Dipende da fattori come la complessità, la lingua di programmazione e lo stile di programmazione.
36. **Eclipse**: Un ambiente di sviluppo integrato (IDE) per varie lingue di programmazione.
37. **Stakeholder**: Tutte le parti interessate nel progetto software, inclusi clienti, utenti finali, sviluppatori, investitori.
38. **Thin client**: Un computer client in una rete che dipende in gran parte dal server centrale per le funzionalità di elaborazione.
39. **Testing**: Risponde alla domanda "Il sistema fa ciò che deve fare?"
40. **Validazione**: Risponde alla domanda "Stiamo costruendo il sistema giusto?"
41. **Affermazioni vere**: Riguardano aspetti come l'accuratezza dei requisiti, l'efficacia della comunicazione e la qualità del prodotto finale.
42. **Black-box testing**: Focus sull'input e l'output senza considerare la struttura interna.
43. **UML (Unified Modeling Language)**: Un linguaggio standardizzato per la modellazione di sistemi software.
44. **Variante**: Si distingue per le differenze in specifiche funzionalità o configurazioni.
45. **Versione**: Si distingue per l'evoluzione nel tempo, con miglioramenti o correzioni di errori.