Agile Software Development

What is "Agility"?

- Effective (rapid and adaptive) response to change (team members, new technology, requirements)
- Effective communication in structure and attitudes among all team members, technological and business people, software engineers and managers.

What is "Agility"?

- Drawing the customer into the team. Eliminate "us and them" attitude. Planning in an uncertain world has its limits and plan must be flexible.
- Organizing a team so that it is in control of the work performed

What is "Agility"?

- Eliminate all but the most essential work products and keep them lean.
- Emphasize an incremental delivery strategy as opposed to intermediate products that gets working software to the customer as rapidly as feasible.

What is "Agility"?

Yielding …

- Rapid, incremental delivery of software
- The development guidelines stress delivery over analysis and design although these activates are not discouraged, and active and continuous communication between developers and customers.

Agile

- "The literal meaning of Agile is "Able to move quickly and easily",
  - in terms of software development, Agile would imply "flexible" process to respond to changes quickly, allowing a developer to go back to a previous stage and carry out necessary changes there by refining the software without much delay as compared to the convention plan driven Software development methods ".

Rationale for Agile

- Agile Methodologies have overcome the traditional methods of waterfall model by becoming flexible, fast, lean, responsive, and consistent.
- Agile method focuses on people and is more communication-oriented
- Agile methods are tested in a dynamic environment and prove to be very flexible by adapting to the change happening in the business.

7

Rationale for Agile

- Agile methods include regular inspection in a disciplined manner, which consequently improves the leadership qualities to boost the teamwork.
- Agile method follows best practices that help in getting high-quality software very quickly.

8

An Agile Process

- Is driven by customer descriptions of what is required (scenarios). Some assumptions:
  - Recognizes that plans are short-lived (some requirements will persist, some will change. Customer priorities will change)

9

An Agile Process

- Develops software iteratively with a heavy emphasis on construction activities (design and construction are interleaved, hard to say how much design is necessary before construction.
- Design models are proven as they are created.)
- Analysis, design, construction and testing are not predictable.

10

An Agile Process

- Thus has to Adapt as changes occur due to unpredictability
- Delivers multiple 'software increments', deliver an operational prototype or portion of an OS to collect customer feedback for adaption.

11

Agile Manifesto

- We are uncovering better ways of developing software by doing it and helping others do it.
- Through this work we have come to value:
  - Individuals and interactions over processes and tools Working software over comprehensive documentation

12

Agile Manifesto

- Customer collaboration over contract negotiation Responding to change over following a plan
- That is, while there is value in the items on the right, we value the items on the left more.

(Kent Beck Et Al)

13

Principles Behind the Manifesto

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

14

Agility Principles - I

4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face–to–face conversation.

15

Agility Principles - II

7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.

16

Agility Principles - II

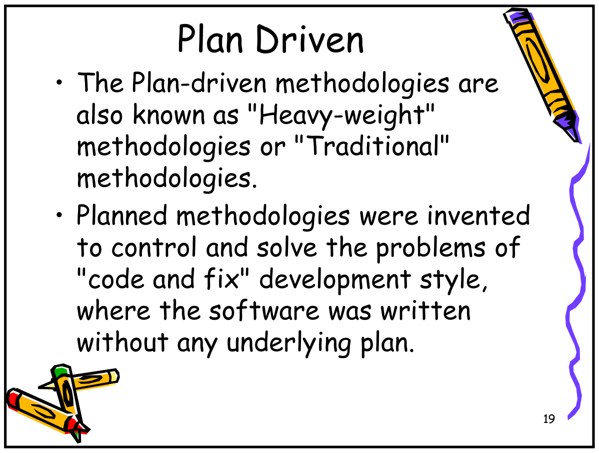
10. Simplicity – the art of maximizing the amount of work not done – is essential.
11. The best architectures, requirements, and designs emerge from self–organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

17

Plan Driven

- Plan-driven software development is a formal and well planned methodology for developing software applications.
- Plan-driven methodologies incorporate repeatability and predictability, well defined incremental process, extensive documentation, up-front system architecture, detailed planning, process monitoring, resource controlling, risk management, verification and validation and user training.

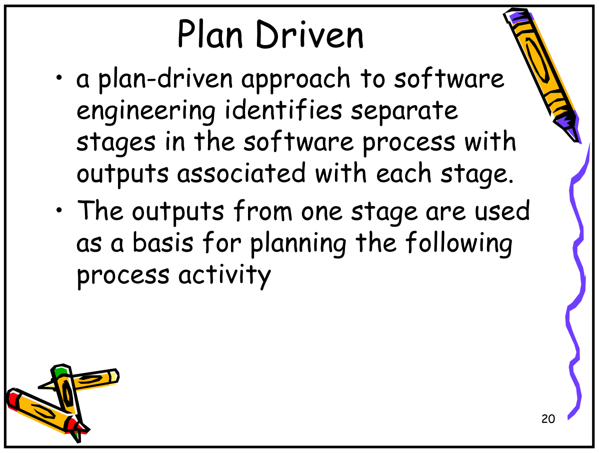
18



Plan Driven

- The Plan-driven methodologies are also known as "Heavy-weight" methodologies or "Traditional" methodologies.
- Planned methodologies were invented to control and solve the problems of "code and fix" development style, where the software was written without any underlying plan.

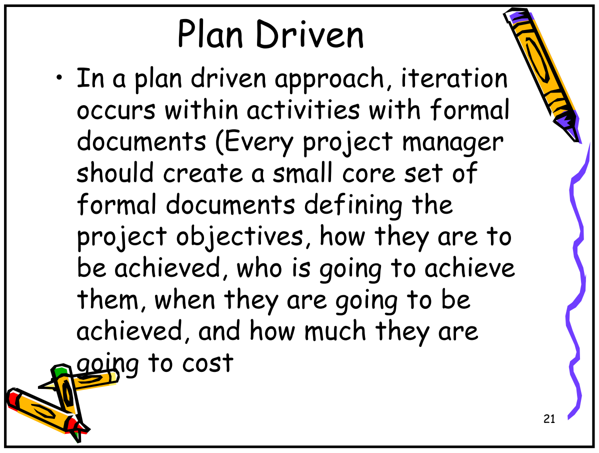
19



Plan Driven

- a plan-driven approach to software engineering identifies separate stages in the software process with outputs associated with each stage.
- The outputs from one stage are used as a basis for planning the following process activity

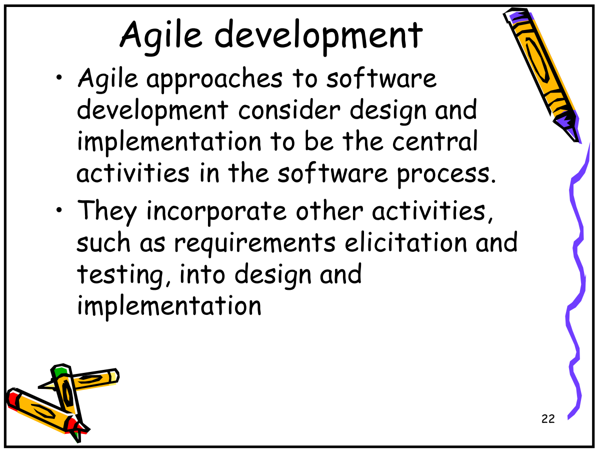
20



Plan Driven

- In a plan driven approach, iteration occurs within activities with formal documents (Every project manager should create a small core set of formal documents defining the project objectives, how they are to be achieved, who is going to achieve them, when they are going to be achieved, and how much they are going to cost

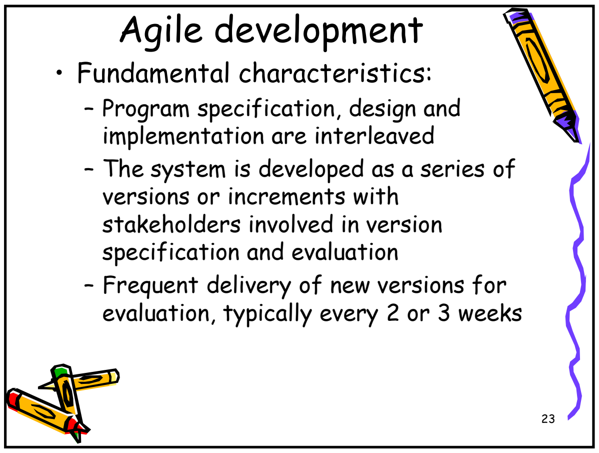
21



Agile development

- Agile approaches to software development consider design and implementation to be the central activities in the software process.
- They incorporate other activities, such as requirements elicitation and testing, into design and implementation

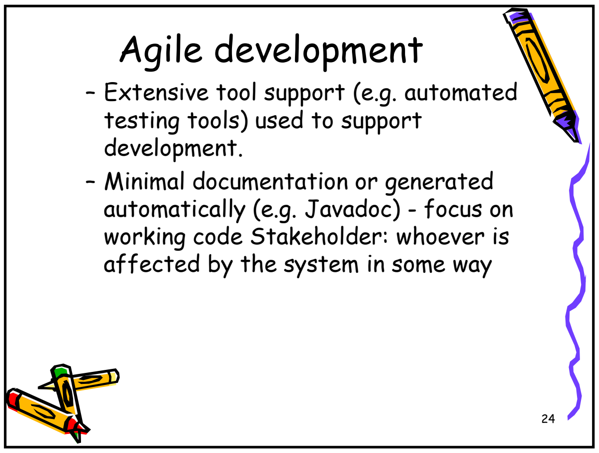
22



Agile development

- Fundamental characteristics:
  - Program specification, design and implementation are interleaved
  - The system is developed as a series of versions or increments with stakeholders involved in version specification and evaluation
  - Frequent delivery of new versions for evaluation, typically every 2 or 3 weeks

23



Agile development

- Extensive tool support (e.g. automated testing tools) used to support development.
- Minimal documentation or generated automatically (e.g. Javadoc) - focus on working code Stakeholder: whoever is affected by the system in some way

24

Plan Driven Vs Agile

- Plan-driven development ▪
  - A plan-driven approach to software engineering is based around separate development stages with the outputs to be produced at each of these stages planned in advance.
  - The outputs from one stage are used as a basis for planning the following process activity.
  -

25

Plan Driven Vs Agile

- Iteration occurs within activities.
- Agile development ▪
  - Specification, design, implementation and testing are interleaved.
  - Iteration occurs across activities

26

Plan Driven Vs Agile

- Most projects include elements of plan-driven and agile processes.
- Deciding the balance depends on Technical, human, organizational issues:

27

Plan Driven Vs Agile

- Is it important to have a very detailed specification and design before moving to implementation? If so, use a plan-driven approach.
- Is an incremental delivery strategy realistic? If so, consider using agile
- How large is the system that is being developed? Use Agile if system is to be developed with a small co-located team.

28

Plan Driven Vs Agile

- What type of system is being developed? Plan-driven may be required for systems that require a lot of analysis before implementation (e.g. real-time system with complex timing requirements).
- What is the expected system lifetime? Long-lifetime systems may require more design documentation to communicate the original intentions of the system developers to the support team.

29

Plan Driven Vs Agile

- What technologies are available to support system development? Agile methods rely on good tools to keep track of an evolving design
- How is the development team organized? If the development team is distributed or if part of the development is being outsourced, then software engineer may need to develop design documents to communicate across the development teams..

30

Plan Driven Vs Agile

- Are there cultural or organizational issues that may affect the system development? Traditional engineering organizations have a culture of plan-based development, as this is the norm/standard in engineering
- How good are the designers and programmers in the development team? It is sometimes argued that agile methods require higher skill levels than plan-based approaches in which programmers simply translate a detailed design into code

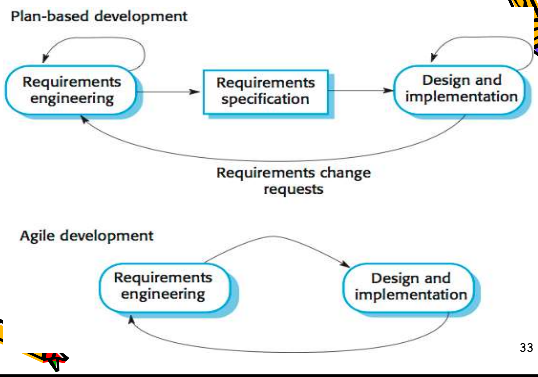
31

Plan Driven Vs Agile

- Is the system subject to external regulation? If a system has to be approved by an external regulator then software engineer will probably be required to produce detailed documentation as part of the system safety case.

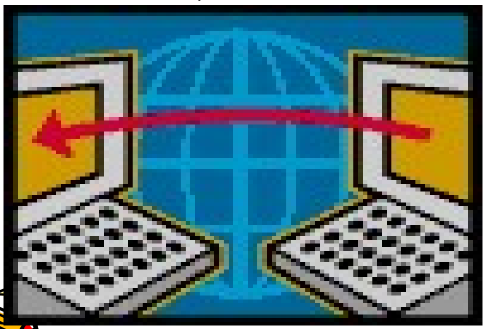
32

Plan Driven Vs Agile


Plan-based development
Requirements engineering
Requirements specification
Design and implementation
Requirements change requests
Agile development
Requirements engineering
Design and implementation

33

Questions



34