Software processes

1. Software processes
   1. Study of software process is the core of software engineering
   2. Good process will give Good product
2. What is a process
   1. Process consists of activities/steps to be carried out in a particular order
   2. Software process
      1. Related with both technical and management issues
   3. Different types of processes
      1. S/D process produces software as end result
      2. Multiple processes may exist
3. Characteristics of a good process
   1. Should be precisely defined
   2. Must be predictable
   3. Also predictable for quality
   4. Should be under statistical control
   5. Should support testing and maintainability
   6. Facilitates early detection of and removal of defect
   7. It should facilitate monitoring and improvement (based on feedback and new tools)
4. Waterfall model
   1. Requirements
   2. Design
   3. Implementation
   4. Transfer to users
   5. Maintenance
5. Waterfall metaphor (linear process)
   1. Use in construction and manufacturing
      1. Collect the requirements
      2. Design
      3. Follow the design
      4. Transfer finished product to the user
      5. Solve problems through maintenance
   2. Intuitively appealing metaphor
      1. Good design avoids the expensive late rework
      2. Waterfall became the dominant paradigms
      3. Unfortunately, requirements always change
6. Waterfall paradigm
   1. Elaborate up-front activities
      1. BDUF (big design up front)
      2. Value may turn out to be small
   2. Textbook
7. Software lifespan, stage model
   1. Initial development
      1. First running version
   2. Evolution
      1. Evolution changes
      2. Software changes
      3. Reengineering
      4. Code decay
   3. Servicing
      1. Servicing patches
      2. Servicing discontinued
   4. Phase-out
      1. Switch-off
   5. Close-down
8. Versioned staged model (V-Model)
   1. Used by software with many users
   2. Evolution is backbone of the process
      1. Evo produce version
      2. Version are serviced, phased-out, closed down
   3. Again, we thus need a process
      1. That supports changes
      2. Develop software with “changes” in mind
9. Variability of processes
   1. The team
      1. Organization, collaboration, skills
   2. The system
      1. Technology, domain, size, and expected quality
10. Granularity
    1. Coarse granularity process
       1. Deal with long periods of time
       2. Software lifecycles are example of a process of very coarse granularity
       3. Stages that are also processes
    2. Word “process” usually used for processes that fit within a single stage or few neighboring stage
11. Granularities of processes
    1. Lifecycle stage
       1. Staged, V-Model, Evolution, servicing
    2. Process
       1. SIP – Solitary Iterative Process
    3. Task
       1. Software change, acceptance testing
    4. Subtask, phase
       1. Concept location, actualization
    5. Step, action
       1. Inspection of a class
12. Forms of process
    1. Process model
       1. Prescription what the tasks should be and how should they fit together – the waterfall model
       2. A blue print how to do things
    2. Enactment
       1. The actual process in the project
       2. Inevitable deviations and exceptions from the process model
    3. Performance
       1. Set of measure that an observer of an enacted process collects
          1. The time, the cost, quality
    4. Plan
       1. Expected future performance
       2. Decisions that the project stakeholders take
          1. Alternatives how to enact the process model
13. Solitary iterative process (SIP)
    1. Single programmer repeats software changes
    2. Functionality is added one step at a time
    3. Repeated changes are the basis of
       1. Software evolution
       2. Software servicing, reengineering
    4. SIP demonstrates characteristics shared by all iterative process
14. Why SIP at all
    1. Why should solitary programmers follow a predefined process?
       1. Rather than flexibly react to challenges
    2. Even the solitary programmers have to meet their obligation
       1. Fulfill their promises, pay their bills
       2. They need to plan the future
       3. They need to manage their own resources
    3. SIP is the process that allows all above
15. Work products
    1. Product backlog
       1. Collect all change requests
       2. Represents the vision for the future of the software
       3. Bugs in the functioning of software
       4. New demands on the software
       5. New ideas what would improve the software
    2. Software code
16. SIP Model
    1. Product backlog
    2. Software changes
    3. Code update
    4. Baseline
    5. Iteration/release
17. Enactment of SIP
    1. BR: new change request arrives
    2. Ini: add cashier session
    3. CL cashier record
    4. IA: estimated set has 4 classes
    5. Ref:
    6. Ex
    7. Act:
    8. Bas
18. Measuring SIP
    1. The data indicate how the process is working
    2. Data serve as a foundation for future planning
19. Time
    1. Total time
    2. Clean time
       1. Clean time = end – start – time of interruptions
20. Log = raw data
    1. Can become large
    2. What can we get from the log
       1. Weekly summary of the clean time
       2. The average time for concept location
       3. Concept location is becoming faster or slower
       4. Recurring exception to the SIP
21. Program size
    1. The complexity of a task is often related to the size of the code programmer deal with
    2. The most common measure is program size
    3. Number of lines of source code
       1. LOC, KLCO, MLOC
    4. This measure is very inaccurate
       1. Different programming language = different size
       2. Different programming style = different size
22. LOC
    1. No proper metrics for measuring program size
    2. LOC is the most commonly used measure of program size
    3. Only the one or two most significant digits are meaningful
       1. 900 LOC, 23 KLOC, 3.2MLOC
    4. Other measures than LOC
       1. Alternative metric of program size
          1. Number of method
          2. Number of classes
          3. Number of files
       2. These are even less accurate than LOC
23. Code defects
    1. Software quality is a major concern
    2. Incorrect computation, premature termination
    3. Defect density
       1. Good quality software: 2.0 defect per KLOC
       2. Poor: higher density
       3. NASA Space Shuttle Avionics software
          1. Defect density estimated 0.1 defects per KLOC
          2. Cutting edge of what can be achieved
24. Defect log
    1. What are the task most likely to introduce a defect?
    2. What is the average time from introduction of a bug to its discovery and fix?
    3. How many known but unfixed defects are in the software on any particular date?
25. Planning
    1. Planning is a prediction of the future
       1. There are uncertainties and risks involved
    2. The data about the past are good predictors about the future
       1. Closely related
       2. Future cannot be predicted with any level of certainty without knowing the past
26. Repetitions
    1. The easiest things to predict are the repetitions
    2. Eliminating the risk and uncertainty is one of the main topics of the planning
       1. Emphasize the respective nature of the process
       2. Repetition is the mother of skill
    3. Unique and unprecedented task are hard to plan
27. Planning software changes
    1. Analogy
       1. Estimate the time needed for the phases
          1. Find similar phase in the past
    2. Decomposition
       1. Decompose the change into phases
       2. Get the sum for all phases
       3. Errors may compensate each other
28. Tasking
    1. Changes should be made more alike
       1. They will be more predictable
    2. Narrow range of size
       1. Epics
       2. Divide large changes into smaller ones
29. Planning baselines
    1. Schedule them at regular intervals
       1. Every day at end of the shift
       2. Every other day after a change is finished
    2. Postponing is not recommended
30. Release Plan
    1. Involve business consideration
    2. Release with a certain new functionality on a certain date
       1. Planning make sure that this promise