Project Management

Tuesday, 4 December 2018 10:10 PM

Revision Control

- Revision Control Software (RCS)
- Managing multiple versions of a piece of information for collaboration
- Revision = version
- Repository = database of history of directory, tracked by RCS (e.g. Git)
  - Stores meta-data about revision history
  - Based in a working directory (HEAD)
  - Stage = prepare for commit (saving current state to Git revision history, can be tagged)
  - Some files can be ignored (e.g. Temp log files)
  - Changes can be stashed
- RCS tool can diff two commits to look at changes, checkout commit to restore it
- Remote Repositories
  - Can be cloned locally
  - Push to and pull form remote repo
  - Fork = copy of remote repo
  - PR = formal request to maintainers of repo to contribute code (GitHub feature)
    - Base fork = where to apply changes, Head fork = contains changes
- Branching: evolving multiple versions in parallel, can be merged (merge commit)
  - Usually, have to switch to master before creating a new branch
  - Merge conflicts: same part of code changed, RCS unsure which change to keep
  - Fast-forward merging (by default): avoids additional merges, moves branch commits to master
  - Rebase: branched move forward as if it was branched later (a more advanced technique, rewrites history)
  - Cherry-picking: create new branch, choose some commits to move over
- Distributed vs Centralised (one central repo)
  - DRCS: Forking workflow
  - CRCS: All changes directly on main repo
    - Feature Branch flow: no forks, only one main remote repo
    - Centralised Flow: no branches, all changes in master

Commit messages

- Imperative mood: "When merged, this commit will..."
- Scope: message (e.g. Scope = developer guide)
- Current state, problems w current state, changes created by commit

Project Planning

- Work Breakdown Structure (WBS): tasks, subtasks, prerequisite tasks/ effort estimates
  - Effort: man hour/ day/ month (md)
  - Tasks should be well-defined: when are they considered done?
- Milestones: significant progress markers (e.g. Intermediate releases)
- Buffers: time set aside to absorb unforeseen delays
  - To help effort estimation, do not inflate task estimates with hidden buffers
  - Buffers are not padding (do not put too much buffer)
- Issue Trackers (e.g. Jira in BitBucket)
- GANTT chart: 2D bar chart of tasks over time
  - Solid = main, grey = subtask, diamond = milestone
- PERT (Program Evaluation Review Technique)
  - Directed graph showing precedence of tasks and effort estimates
  - Critical path: affects shortest possible delivery time

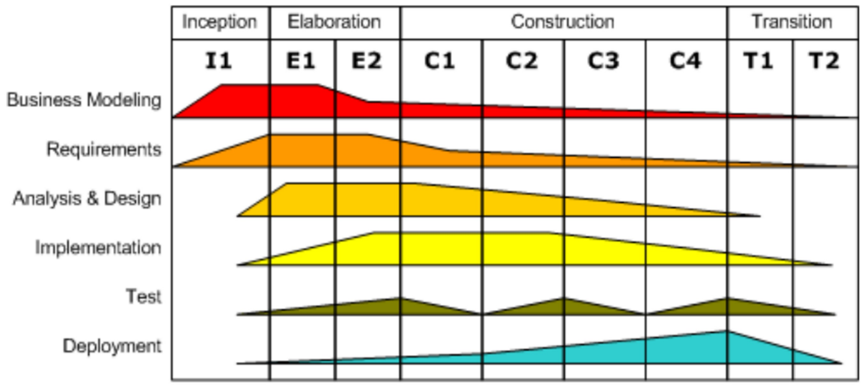
Teamwork

- Team Structures: egoless, chief-programmer, strict-hierarchy
  - Responsibility and accountability
- Analysts (user-end) -> Designers -> Engineers (implement)

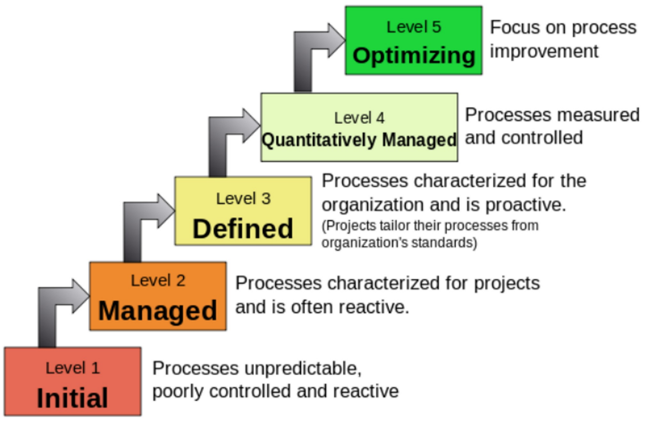
SDLC Process Models

- Requirement -> Analysis -> Design -> Implementation -> Testing
- Sequential (waterfall): linear SDLC
  - Useful when problem statement is well-understood and stable
  - Project organised based on activities
- Iterative (incremental): numerous iterations
  - Requirements changing over time
  - Project organised based on functionality
  - Breadth-first: major components in parallel
  - Depth-first: some components fleshed out
- Agile manifesto (vs full design: changes hard to make)
  - Individuals and interactions > processes and tools
  - Working software > comprehensive documentation
  - Customer collaboration > contract negotiation
  - Responding to change > following plan
  - Examples: eXtreme Programming (XP) and Scrum
  - Pros:
    - More focus on customer satisfaction
    - Less chance of building the wrong product (because of frequent customer feedback)
    - Less resource wasted on bureaucracy, over-documenting, contract negotiations
  - Cons:
    - It is 'just hacking'. Not very systematic. No discipline
    - It is hard to know in advance the exact final product
    - It does not give enough attention to documentation
    - Lack of management control (gives too much freedom to developers)
- XP: stresses customer satisfaction (delivers software as customer needs it)
  - Empower developers to confidently respond to changing customer requirements
  - Emphasis on teamwork
  - Improve software project communication, simplicity, feedback, respect, courage
  - Pair programming, CRC (class-responsibility-collaboration) cards, project velocity (measure progress using estimates of user stories and tasks completed), standup meetings (quick status updates)
    - Pros:
      - Better quality code (>1 person knows about any piece of code)
      - Learn from each other (train new programmers)
      - Better discipline, time management and morale
    - Cons:
      - Increase in total man hours required
      - Personality clashes between pair-members
      - Workspaces need to be adapted(two people one computer)
      - Rotation: one developer needs to know multiple parts
- Scrum: a set of practices and roles
  - Scrum Master: maintains processes
  - Product Owner: stakeholders and business
  - Team: cross-functional group involved in whole SDLC
  - Based on product backlog (high level requirements of work to be done)
  - Planning meeting -> Iterations = sprints (1wk to 1mth) -> Review/ retrospective meeting
    - Controlled by sprint backlog (frozen during sprint)
  - Requirements churn: customers change what they need and want

    - 15 min Daily Scrum: what did you do, what you will do, potential problems
- Unified Process (Three Amigos - Ivar Jacobson, Grady Booch, James Rumbaugh)
    - Inception -> Elaboration -> Construction -> Transition



    - Flexible and customisable model framework
    - Can be iterative and incremental
- CMMI: Capability Maturity Model Integration
    - Maturity levels and criteria


Level 5
Optimizing
Focus on process improvement
Level 4
Quantitatively Managed
Processes measured and controlled
Level 3
Defined
Processes characterized for the organization and is proactive.
(Projects tailor their processes from organization's standards)
Level 2
Managed
Processes characterized for projects and is often reactive.
Level 1
Initial
Processes unpredictable, poorly controlled and reactive

- Good, cheap, fast: choose 2