Chapter 28: Managing Construction

Encouraging Good Coding

- Coding is the primary output of construction haha
- Mandating strict set of technical standards isn't a good idea
- If there are to be standards, an architect needs to design them
    - Programmers don't respect their managers
    - Make sure person defining standards is very in touch with current coding tool landscape and wont lose respect for commanding an arbitrary standard

Techniques for Encouraging Good Coding

- Assign two people to every part of the project
    - At least two people will think the code works and is readable
- Review every line of code
- **Require code sign offs**
    - **Like stamping technical drawings**
- Route good code examples for review
- Emphasize that code is a public asset
    - Don't let people think their code is "their code"
- Reward good code
    - Rewards should be things that the programmer wants
        - Not just "attaboy"
    - Code that receives an award should be exceptionally good
        - DO NOT want to reward a developer that everyone knows does shit work
- Come up with one easy standard
    - Like the manager must be able to read and understand the code
    - No clever code

Configuration Management

What is Configuration Management

- The practice of identifying project artifacts and handling changes systematically
    - Main goal is preserving the integrity of the project over time and through changes
- Need to control how changes are made to requirements
- Software Configuration Management
    - SCM

Requirements and Design Changes

- Ideas are cheap, need to put gates on ideas attempting to be implemented
- Follow a systematic change control procedure
- Handle change requests in groups

- Estimate the cost of each change
- Be wary of high change volumes
    - Sign of poor requirements, architecture, or top level designs
- Establish a change control board
- Watch for bureaucracy but don't let the fear of it effect change control
    - Need to streamline without letting things get off the rails

Software Code Changes

- Version control software
    - Just use it
    - Make sure to make nots on why and what you changed
    - Benefits
        - Don't step on anyone's toes
        - Easily update all versions of code
        - Backtrack to old versions
        - Don't have to worry about backups

Machine Configurations

- Use standard image
- Update regularly as needed

Backup Plan

- Backup work and configurations periodically
- Do good prep work to archive projects too

Estimating a Construction Schedule

Estimation Approaches

- Use estimating software
- Use an algorithmic approach
    - Like Cocomo II
- Have outside estimation experts estimate the project
- Have a walk through meeting for estimates
- Estimate pieces of the project, then add the pieces together
- Have people estimate their own tasks, then add the task estimates together
- Refer to experience on previous projects
- Keep previous estimates and see how accurate they were
    - Use them to adjust new estimates
- Establish objectives
    - Why do you need an estimate?
    - What are you estimating
    - Are you including vacation time?
    - How certain does the estimate need to be?

- Allow time for the estimate and plan it out
    - DO NOT RUSH IT
- Need to know software requirements first
- Estimate at a low level of detail
- Use multiple techniques and compare results
- Re-estimate periodically

Estimating the Amount of Construction

- Only reasonable data is data collected by an organization on past projects

Influences on Schedule

- Multi site development is 22% detrimental
- Project complexity is 74% detrimental
- Requirements analyst capability is 42% detrimental
- Storage constraints are 46% detrimental
- Time constraints are 63%
- There are a bunch of other ones

Estimation vs Control

- Estimation is figuring out what it'll take to get done
- Control is how you actually manage and allocate folks to get it done

What to Do If You're Behind

- Pretty much every project is gonna be late haha
- Late projects only get more late haha
- Expanding the team doesn't help either
    - Maybe start with more folks than you think the project needs
- **Reducing the scope of the project actually helps**
- **Re-estimate the development time for the least important features actually helps**

Measurement

- People tend to focus on work that's measured and to ignore work that isn't
- Things to measure
    - Size
        - Total lines of code written
        - Total comment lines
        - Total number of classes or routines
        - Total data declarations
        - Total blank lines
    - Defect Tracking
        - Severity of each defect
        - Location of each defect (class or routine)
        - Origin of each defect (requirements, design construction, test)
        - Way in which each defect is corrected

        - Person responsible for each defect
        - Number of lines affected by each defect correction
        - Work hours spent correcting each defect
        - Average time required to find a defect
        - Average time required to fix a defect
        - Number of attempts made to correct each defect
        - Number of new errors resulting from defect correction
    - Productivity
        - Work-hours spent on the project
        - Work-hours spent on each class or routine
        - Number of times each class or routine changed
        - Dollars spent on project
        - Dollars spent per line of code
        - Dollars spent per defect
    - Overall quality
        - Total number of defects
        - Number of defects in each class or routine
        - Average defects per thousand lines of code
        - Mean time between failures
    - Maintainability
        - Number of public routines on each class
        - Number of parameters passed to each routine
        - Number of private routines and or variables on each class
        - Number of local variable used by each routine
        - Number of routines called by each class or routine
        - Number of decision points in each routine
        - Control flow complexity in each routine
        - Lines of code in each class or routine
        - Lines of comments in each class or routine
        - Number of data declarations in each class or routine
        - Number of blank lines in each class or routine
        - Number of input or output statements in each class or routine
- Can use software to do a lot of this
- Don't start out by collecting everything lel will get buried in data
- Make sure you're collecting data for a good reason
    - Set goals and determine questions that need to be asked to meet the goals and then measure those answers

Treating Programmers as People

- Religious issues
    - Programming language
    - Indentation styles
    - IDE choice
    - Commenting style

    - Efficiency vs readability tradeoffs
    - Methodology
        - Scrum, EP, evolutionary delivery, etc
    - Programming utilities
    - Naming conventions
    - Global variables
    - Performance measurements
    - **Avoid setting "rules" or "standards", just use "suggestions" or "guidelines"**
- Physical Environment
    - Programmers performing in the top 25% had
        - Bigger
        - Quieter
        - More private offices
        - Fewer interruptions from people and phone calls

Managing Your Manager

- Main thing is need to tell them what to do while tricking them into thinking they are the ones in control
- Some approaches
    - Plant ideas for what you want to do, then wait for manager to have a brainstorm
    - Educate manager on the right way to do things
    - Focus on managers interests, and do what they want you to do. Don't distract them with unnecessary implementation details
    - Refuse to do what manager tells you and insist on doing job the right way
    - Find another job (lmao these savage)