PURDUE UNIVERSITY

Implementation & Testing Plan

CS 307: Software Engineering

Pascal Meunier

Why An Implementation Plan

- If you don't think about how you're going to do it, it's likely going to be:
  - Haphazard
  - Unmanageable
  - Costly
  - Difficult to plan for the release
  - How will you know that it was done properly, in budget, in time, and if not, why?

What's An Implementation Plan?

- See parts of Chapter 11, "Managing the software process"
- Which development process did you decide to use?
- Which features have priority?
- Which ones will you implement, and in which order?
  - Dependencies? Needed resources?

The Capability Maturity Model

- p. 415 of the book
- One of several process standards
- About managing software processes
  - Arguably the main difference between artisanal software development and software engineering
  - Important topic

CMM Levels

1. Ad-Hoc (chaotic, heroics)
   - Most students
2. Repeatable
3. Defined
   - Ideally where we would like you to operate by the end of this class
4. Managed
5. Optimized

Level 3- Defined

- Defined and documented standard processes
  - Some improvement of processes over time
- Consistency across an organization
- Mandatory process objectives

Meta-Programming: Can You Define

- How you will plan your team work
  - How do you divide the work?
  - How do you verify its quality?
- How you will plan the builds
- How you will plan the demo
- Not the plans themselves, but how you go about planning

Version Control

- Are there going to be any automated checks
  - When committing code
    - e.g., parsing check (syntax)
  - Automated nightly build
- Does every commit need to be explained?
  - Every change linked to a recorded bug?

More...

- How have you decided to:
  - Keep track of
    - Bugs
    - Progress
  - Make predictions about delivery dates
  - How are you ranking the features in order of priority
    - Then which features will you implement in Build 1? Build 2? Extra ones if you happen to have time?

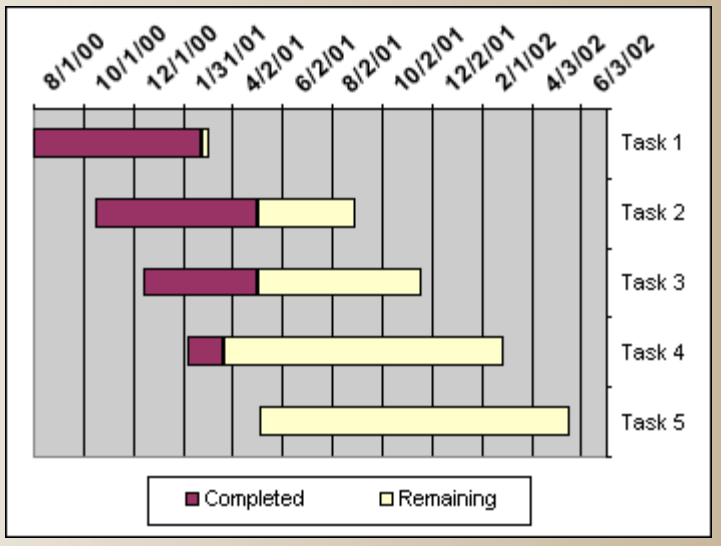
Tools

- Charts
  - PERT
  - Gantt
- Spreadsheet completion estimates

Task Name	Original Esti-mate (hours)	Current Esti-mate	Accom-plished	Started On	Estimated Completion Date
Task A					
Task B					
Task C					

Gantt Chart

- Example (using MS Excel, from MS's web site)



Your Charts

- You would decompose Build 1 into
  - Implementing components
    - Sub-components
      - Classes
      - Tests
- Your estimated times will be wrong
  - You get better with time and recorded history
  - Better than having no idea at all

Implementation Plan

- What we want to see:
  - Define the processes you will use
  - Define the management tools you will use
    - e.g., Gantt chart, spreadsheet table
    - Others
      - e.g., Gnome Time Tracker (Linux)
  - Explain your choices

Suggestion: Risk Driven Approach

- Idea: by taking on the most risky parts first, the schedule stabilizes faster or design is revised earlier
- Expose as early as possible things that
  - Are unrealistic (requirements)
  - Won't work as designed
- Minimize cost of changing directions

Example Risks

- Address show-stoppers first
  - Part of the development environment doesn't work (likely!)
  - We don't know how to install or use language/framework/library X
  - Y isn't fully compatible with Z
    - Or lots of configuration work necessary
  - Will our algorithm really work?

Example Risks

- User interface dependence on how some features are implemented (options)
  - Can't work on user interface until resolved, or change is costly
- Errors in interfaces between modules
  - A change in a module requires a change in another

Identify Risks

- What are the risks to your project and how can you mitigate them?
- Can you work on Z while a solution to X is found?
  - Can you plan your project from the start so that if X is delayed, Z exists?
- Have maneuvering room
- Have a plan B
  - What will you sacrifice?

Resource Management

- Risk: the resource can take a long time to obtain and you can't code Y without it
  - Solution: setup or ask for the resource earlier (ASAP?)
  - Have an alternative resource, maybe not as good, but that allows you to produce valid code
- You are in charge of resources

Schedule and Milestones

- Tasks
- Components vs builds#
- Testing
- Your schedule can be tailored to minimize some risks

Testing (QA) Plan

- Define
  - Tests
  - Metrics
    - Code coverage from tests
    - Tools for metrics
      - Code scanners
  - Other tools
    - Vulnerability scanners

Risk-based Testing

- Prioritize features and functions to be tested based on:
  - Importance
  - Likelihood or impact of failure
- "Good enough software"

Some Types of Testing

- Inline testing
  - Assert macros & statements
- Unit testing
  - Classes, methods, libraries
- Integration testing
  - Interactions between components
- Final testing
  - Exercise entire product

Notion: Test Coverage

- Identify a test coverage tool appropriate for your project
  - Will identify % of code exercised by your tests
- No execution, no bug found
- Complete coverage requires a lot of work and is almost never done
  - Satisfied with some %

Notion: Static Analysis

- Examine the code without executing it
- Common approach:
  - Compile the code to some intermediate representation
  - Analyze
    - Dead (unreachable) code
    - Code quality metrics (complexity)
    - Things that look like vulnerabilities

Contents of Test Plan

- Complete set of test cases
- Testing process
- Refer to chapter 10 of the book

Complete Document

- Merge testing plan template with the outline of a project plan
  - See book, section 11.6
- Base schedule on testing plan, so do plan testing first

Questions?