Project plan debriefing

- Status reporting and configuration management
- Defect reporting
- Configuration management

Agenda for today

1. Quick comments on revised project plans
2. Team introductions and brief description of proposed work (no more than 3 minutes per team)
3. Progress, status, and defect reporting
4. Configuration management
5. Issues related to completion of reqts. specs

Recall...

As a client/manager, I need:
- Evidence that what is being proposed is feasible
- Visibility into your development process so that I can judge whether we will make deadlines

Some plans were more detailed than others

In either case:
- you must follow your plan to the letter, and
- I will use your plan to track your progress

Measuring time on task

Time measurement/management is a critical component of any formal software process

PSP requires you to:
- Estimate time required for each task in your plan
- Accurately measure amount of time spent on each task

Suggest keeping a project notebook (paper):
- As you begin work on a task, list the task identifier and start time (precisely, to the nearest minute)
- If phone rings, record interruption time and then resumption time when you resume
- Time measurements should include total number of minutes on task (i.e., do not report any time spent on other activities)

Henceforth

You MUST accurately track the time you spend on each task!!!

At the beginning of each class, we will spend 10-15 minutes on oral status reports

Specifically, your team must answer:
- What task(s), which were listed in your project plan, have you worked on since last report?
- How much time (hours) did you spend on each task?
- Have task outputs been committed to version control?
- What other (unforeseen) tasks did you spend your time working on? How much time?
- How is the proposed schedule affected?

To simplify reporting...

Every task you perform and every document you produce should be uniquely numbered and versioned

Critical requirement: At any point in time, should be easy to retrieve any version of any document you have ever produced

Suggestions:
- name/number every task in your project plan
- construct a document repository
- construct a web page to index into this repository
Versioning

As we progress, you will find it necessary to revise plans and update artifacts
– E.g., having identified names and unique numbers for each document, project plan should be edited to include these numbers and perhaps even link to the documents themselves
– E.g., as we begin to run out of time, you will need to edit the plan to adjust schedule or scale back on the deliverables

Henceforth

You MUST accurately record and track defects as soon as they are discovered
– Note that defects may be detected prior to having written a single line of code
– E.g., vague or inconsistent requirement, which is only discovered during design or implementation

For every defect found, you must:
– assign a number that uniquely identifies defect
– record:
  • name/number of task you were performing when defect was discovered
  • source (i.e., where and in which task input) of the defect
  • brief description of the defect

Example

Suppose, during design, you discover an ambiguity in one of the requirements
E.g., in text-formatting problem:
– “Input is a sequence of lines ... A line is a sequence of characters that appear between two new-line characters.”
– Question: Must the first and last characters in the input always be a new-line character?

Need to record this requirements defect
– List enough information to allow reader to find the occurrence of the problem
  • e.g., lines 22 and 38 on page 10 of document reqtsSpec-002
  • Describe the two or more interpretations that demonstrate that this is indeed a defect

Fixing defects

Some defects will not require immediate attention; whereas others will
– E.g., ambiguous requirement with low impact on design, vs.
– inconsistent requirement that hijacks design
You are likely to uncover some of the latter type
When this happens, you must:
– Plan and execute a new task for resolving the ambiguity
– Task should be numbered uniquely from the task you were performing when you discovered the defect

Postmortem phase will review this data to determine root cause of defect injection

To simplify defect reporting/tracking

Create a web-based repository of the defect reports
Create an html template to use in defect reporting

Configuration management
Configuration management

Configuration: set of components that, together, constitute a software product
- We refer to each component as a software configuration item (SCI).
- SCIs may be constructed by a developer or derived from other SCIs.
- Familiar examples: .c files, .o files, test cases
- Less familiar examples: reqts spec, defect report

Configuration management involves:
- coordinating development among multiple contributors
- controlling change/evolution of software products

Identification

Each SCI must be uniquely identified (named) and related to dependent SCIs
Identification involves:
- document formats and content
- control procedures for accessing/editing documents
- hierarchy, i.e., what documents are parts of which others?

Derivation involves:
- tools or processes that derive one SCI from others
- input parameters and options
- documentation of purpose

May result in a document or something like a Makefile

Configuration mgmt: Major parts

Identification of SCIs and their dependencies
- E.g., naming/numbering documents

Version control

Change control

Auditing and reporting

Make facility

Simple scheme for version control

Low tech: Construct a hierarchy of directories, one for each document:
- as new versions of the document evolve, make copies in the appropriate directory
- always name the files to include their document/version number
- be sure document/version number appears on first page of document