Project Planning - Chapter 5

Objectives - Enable management and technical personnel to:

⇒ comprehend the problem and solution strategy

⇒ make reasonable estimates of resources needed

⇒ make the decisions to get and keep the project on track
  ⇒ risk analysis - chapter 6
  ⇒ setup initial scheduling and tracking - chapter 7
  ⇒ establish software configuration management - chapter 9
The Problem Definition

To some extent ignored by Pressman

*DO SO AT GREAT PERIL*

⇒ Scope
  ⇒ Justification
    ⇒ Why? - what critical problem(s) will a successful project overcome
    ⇒ Who? - the customer and user
  ⇒ Approach
    ⇒ Feasibility - previous solutions & failures
    ⇒ Solution(s) - preliminary design & analysis to study potential solution(s) - select or combine the best
    ⇒ Understand the critical factors (new tech., new features, etc.) - hard parts
Goal - to get needs committed early

⇒ Adding resources to recover from project failure is costly

⇒ Human
  ⇒ Commitment, experience, environment, support, etc.

⇒ Estimation (Hind sight is 20/20 but costly)
  ⇒ Base on comparable projects, experience
  ⇒ Decompose and estimate subproject costs
    ⇒ Function point - chapter 4
    ⇒ Preliminary analysis - pseudocoding - LOC
  ⇒ Process or object based
⇒ Use empirical software estimation models
⇒ Make/buy decision
Relate to specs, designs, code, test data, etc.

⇒ Off-the-shelf components
  ⇒ existing software developed internally or acquired from third party
  ⇒ care must be taken that they “fit” the project needs

⇒ Full-experience or partial-experience components
  ⇒ Tested and proven
  ⇒ Depending upon the experience with and previous application to?
  ⇒ Components can provide an excellent start. Need to understand modifications
New components

⇒ This is by far the highest risk decision

⇒ In many projects it may be wise to do preliminary design and development of new components to demonstrate their feasibility and capability.

⇒ Avoid the “not invented here” syndrome
Project Estimation

Based on the use of project metrics

1. Subdivide project into “logical” units (LU)
   - Function points, object points, code units with LOC estimates, etc.
2. Weight the difficulty of each
   - May use optimistic (Sopt), likely (Sm) and pessimistic (Spst) and use formula
     \[ S = \frac{Sopt + 4Sm + Spst}{6} \]
   - Use past histories (if available) or guess
3. Sum logical unit estimates
4. Use Estimation model equations \[ E = A \times B(\Sigma(LU)) \]
   - Can be done with different models for subunits
5. If desired repeat with another LU to refine estimates
6. Compare results to past project histories
Modifications and tailoring

⇒ The logical units can be anything which makes sense to the project and people
  ⇒ My guess is that large software houses, e.g., like Oracles, have proprietary LUs with which they have had a world of experience.
⇒ Makes estimates during project stages
  ⇒ COCOMO model of Boehm
    ⇒ Use during initial stage (prototyping), early stage development (architecture established) and construction.
    ⇒ Can give good refinement and feedback to estimates
⇒ Correlate estimates with finished projects - feedback
⇒ Estimate different things
  ⇒ Cost, schedule, personnel needed (possibly by type), etc.
Dicotomy - NIH vs Difficult to correct deficiencies

⇒ Not invented here (NIH)
  ⇒ Tend to over estimate “your” capabilities and experience
  ⇒ Always looks like a piece of cake before start

⇒ Buy
  ⇒ Looks good until some “gotcha” occurs
  ⇒ “Gotcha” hard and costly to correct
    ⇒ In house - source code and a detailed understanding of the program. The latter may be impossible without vendor’s aid.
  ⇒ Out-of-house - Vendor becomes a single source supplier with no negotiating possibilities (buy in)
Risk Analysis - Chapter 6

Cover only briefly

⇒ A risk - recognition that an event may cause change in the normal or anticipated course. What if?
⇒ Risk is an every-day-of-life situation

⇒ Risk occurrence may be very difficult and potentially costly

⇒ Pressman does not cover the down side of risk analysis
  ⇒ Mitigation can be expensive

⇒ Justification for analysis and mitigation can be very subtle.

⇒ Obviously the idea of “Be Prepared” is akin to motherhood
Risk Analysis

⇒ Reactive vs Proactive (unplanned vs planned)
⇒ Sorry there is no such thing as reactive or unplanned risk. You can realize that a certain course of action is risky and decide to live with its consequences and make the necessary adjustments only after it has occurred.

⇒ Risk Categories
⇒ Project
  ⇒ Cost increase, delivery time slippage, staffing, etc.
⇒ Technical
  ⇒ Problem (or subproblem) more difficult to do than original estimate
⇒ Business
  ⇒ Produce not needed; obsolete, difficulties in use; loss support, resources, personnel; image (face); etc.
Risk identification, projection and assessment

⇒ Identified risks - should be handled in some manner as soon as they become apparent

⇒ To plan for unforeseen problems:
  ⇒ Add or incorporate a factor in weightings
    ⇒ Increase weighting from (3) optimistic, probable, pessimistic to (4) catastrophic, critical, marginal, negligible
  ⇒ Add a contingency to plans to get a range of estimates

⇒ Subdivide problem further. Hopefully subdivision may enable the problem to become anticipated
Risk mitigation

For Important Risks

⇒ Develop and cost out alternative strategy(s)

⇒ Keep close watch over the project to ascertain whether and when an alternative strategy should be initiated
   ⇒ Devise and implement watch actions and metrics

⇒ Continue close observation to see if either strategy should be terminated.
   ⇒ Continue watch actions and metrics

Prompt and correct action is extremely important. It can be the key to success
A very important part of project management planning

⇒ Provides for an intelligent progression of project subtasks
⇒ Permits use of the divide and conquer philosophy
⇒ Allows visibility into the project’s progress - late subtasks, problems, etc.
⇒ Allows for the intelligent and timely redirection of resources

⇒ Provides insight into the inter-dependence of subtasks
⇒ Provides for the reasonable interfacing and integration of subtasks
Project Scheduling (cont’d)

Basics

⇒ Compartmentalization - same principle as subdivision
  ⇒ The subdivisions used for metrics and project planning (FP, OP or any other logic breakdown of the work)
  ⇒ Same breakdown as planning is an advantage because costs and time are highly correlated

⇒ Time & effort allocations - Determine the time & effort (resources) needed for each task

⇒ Interdependency & Independent - the relationship between tasks
  ⇒ What tasks must be completed (or far along) before other tasks can begin? What can be overlapped?
  ⇒ What subtasks must be grouped for test and debug?
Project Scheduling (cont’d)

Basics (cont’d)

⇒ Milestones
  ⇒ Each subtask must identify a series of sequential events (observables) which measure task progress
  ⇒ The time to reach each milestone needs to be estimated
  ⇒ Note this further subdivides tasks
  ⇒ Obviously - missed milestones (unless readjusted) are a prime indication of potential trouble.

⇒ Deliverables
  ⇒ Outcome and output of each subtask
  ⇒ May not only include software but documentation, test and performance results
Project Scheduling

⇒ Timeline chart
  ⇒ Premier scheduling document for the project
  ⇒ Show subtask (possibly at more than one level)
  ⇒ Shows identified milestones
  ⇒ Note - unlike Figure 7.4 subtask activity may overlap

⇒ Also or alternatively PERT and/or Critical Path charts can be used
  ⇒ PERT - a directed graph of subtasks with time as the x-axis
  ⇒ Shows subtask dependency so estimate of project time and interconnectivity
  ⇒ Shows critical path longest route
  ⇒ Usually not as detailed task wise as timeline chart
A proposal to NSF for network research

H3M Project time schedule

- Simulator development
- Media access
- BW man & load bal integration
- Performance and scalability studies
  - BW man & load bal studies
  - Protocol ops and reliability
  - Interoperability studies
- Testbed dev.

Time in quarters: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
Project Scheduling

A critical activity in project planning

⇒ All RFP’s (Request for Proposal) require some form scheduling information, usually Timeline charts

⇒ Schedule monitoring throughout the project is necessary
⇒ the primary mechanism for spotting potential and real problems

⇒ Updating the project schedule is necessary to maintain a correct view
Organize activity to best assure “good” software

⇒ In organizations where software is the life-blood of the company SQA is important

⇒ Like any company, defective products may cause demise

⇒ Usually done as a company wide plan where every aspect of the software development process is examined and improvements made to overcome weaknesses

⇒ Obviously, software metrics play a big part in defining problems and the solutions

⇒ SQA may be approached on a problem basis.
⇒ Plan - See the list in text on specific actions
SQA vs Risk Management (revisited)

Mistake proof software - section 8.9

⇒ I would put this concept under risk mitigation
  ⇒ the concept is much older than 1960
  ⇒ in many situations it was called an interconnect - the prevention or restriction of an action based upon state
⇒ Examples
  ⇒ The "hill holder" in early Nash cars
  ⇒ Menu item dimming when action is not possible
  ⇒ Restricted rudder range in non-landing situations. Spoiler locks until touchdown.
⇒ Consider risks from a product development and product use view point
⇒ In today’s litigious society seriously consider product use risk mitigation
An important aspect for software

⇒ Versions have existed in systems for many years

⇒ Automobiles by model year

⇒ Airplanes due to modifications & improvements

⇒ Incorporate new utility or new technology

⇒ Change is a way of life
Configuration management structure

When a product has reached a certain stage of maturity

⇒ Baseline
  ⇒ A product whose configuration has been reviewed and agreed upon to serve as a basis for further development
  ⇒ Usually marked by delivery or release of first product
  ⇒ Documentation is in place
⇒ Software Configuration Items
  ⇒ Change items, redemdes, etc. - identify
⇒ Stages
  ⇒ Go through the process - specification, requirements, design, development, test, integration, etc.
  ⇒ Approve change - type depends upon organization
⇒ Integrate with baseline & store in project data base
Config. Manage. Identification

A data structure, storage and identification problem
⇒ Structure of product
  ⇒ Basic and Aggregate Items
  ⇒ Each aggregate item lists its basic items and provides a link to each
  ⇒ Each item has a description, text, drawings, etc. whatever is needed to document its existence and features.
  ⇒ Give each item a unique version number

⇒ Add Change Item - create a new version number
  ⇒ Augment each item’s documentation which change documents (separately but linked)
  ⇒ Provide links - depends upon nature of future product search requirements
    ⇒ directly through change
    ⇒ between baseline and change
Configuration Management (cont’d)

Change made easier - client-server object models

Note: queries and results at client must remain fixed under change

Client

Queries & results fixed

Added queries & results

Server

Interface Provides queries & actions

Added Interface Provides queries & actions

New Object code and data do requested actions