Software Planning and Estimation Process : A Review

Shivneet Singh

Abstract : Software planning involves estimating how much time, effort, money, and resources will be required to build a specific software system. After the project scope is determined and the problem is decomposed into smaller problems, software managers use historical project data (as well as personal experience and intuition) to determine estimates for each. The final estimates are typically adjusted by taking project complexity and risk into account. The resulting work product is called a project management plan. Managers will not know that they have done a good job estimating until the project post mortem. It is essential to track resources and revise estimates as a project progresses.

ISSN 2454-308X

Key Words : Software Planning and Estimation Process

Project Planning Objectives :

- To provide a framework that enables software manager to make a reasonable estimate of resources, cost, and schedule.
- ‘Best case’ and ‘worst case’ scenarios should be used to bound project outcomes.
- Estimates should be updated as the project progresses.

Estimation Reliability Factors

- Project complexity
- Project size
- Degree of structural uncertainty (degree to which requirements have solidified, the ease with which functions can be compartmentalized, and the hierarchical nature of the information processed)
- Availability of historical information

Project Planning Process

1. Establish project scope
2. Determine feasibility
3. Analyze risks
4. Determine requires resources
    a. Determine required human resources
    b. Define reusable software resources
    c. Identify environmental resources
5. Estimate cost and effort

   a. Decompose the problem
   b. Develop two or more estimates
   c. Reconcile the estimates
6. Develop project schedule
   a. Establish a meaningful task set
   b. Define task network
   c. Use scheduling tools to develop timeline chart
   d. Define schedule tracking mechanisms

Software Scope

- Describes the data to be processed and produced, control parameters, function, performance, constraints, external interfaces, and reliability.
- Often functions described in the software scope statement are refined to allow for better estimates of cost and schedule.

Customer Communication and Scope

- Determine the customer's overall goals for the proposed system and any expected benefits.
- Determine the customer's perceptions concerning the nature if a good solution to the problem.
- Evaluate the effectiveness of the customer meeting.

Feasibility

- Technical feasibility is not a good enough reason to build a product.
- The product must meet the customer's needs and not be available as an off-the-shelf purchase.

Estimation of Resources

- Human Resources (number of people required and skills needed to complete the development project)
- Reusable Software Resources (off-the-shelf components, full-experience components, partial-experience components, new components)
- Environment Resources (hardware and software required to be accessible by software team during the development process)

Software Project Estimation Options

- Delay estimation until late in the project.
- Base estimates on similar projects already completed.
- Use simple decomposition techniques to estimate project cost and effort.
- Use empirical models for software cost and effort estimation.

- Automated tools may assist with project decomposition and estimation.

Decomposition Techniques

- Software sizing (fuzzy logic, function point, standard component, change)
- Problem-based estimation (using LOC decomposition focuses on software functions, using FP decomposition focuses on information domain characteristics)
- Process-based estimation (decomposition based on tasks required to complete the software process framework)
- Use-case estimation (promising, but controversial due to lack of standardization of use cases)

Causes of Estimation Reconciliation Problems

- Project scope is not adequately understood or misinterpreted by planner
- Productivity data used for problem-based estimation techniques is inappropriate or obsolete for the application

Empirical Estimation Models

- Typically derived from regression analysis of historical software project data with estimated person-months as the dependent variable and KLOC, FP, or object points as independent variables.
- Constructive Cost Model (COCOMO) is an example of a static estimation model.
- COCOMO II is a hierarchy of estimation models that take the process phase into account making it more of dynamic estimation model.
- The Software Equation is an example of a dynamic estimation model.

Estimation for Object-Oriented Projects

1. Develop estimates using effort decomposition, FP analysis, and any other method that is applicable for conventional applications.
2. Using the requirements model (Chapter 6), develop use cases and determine a count. Recognize that the number of use cases may change as the project progresses.
3. From the requirements model, determine the number of key classes (called analysis classes in Chapter 6).
4. Categorize the type of interface for the application and develop a multiplier for support classes
5. Multiply the number of key classes (step 3) by the multiplier to obtain an estimate for the number of support classes.
6. Multiply the total number of classes (key + support) by the average number of work-units per class.

7. Cross check the class-based estimate by multiplying the average number of work-units per use case.

Estimation for Agile Development

1. Each user scenario is considered separately
2. The scenario is decomposed into a set of engineering tasks
3. Each task is estimated separately
    a. May use historical data, empirical model, or experience
    b. Scenario volume can be estimated (LOC, FP, use-case count, etc.)
4. Total scenario estimate computed
    a. Sum estimates for each task
    b. Translate volume estimate to effort using historical dsata
5. The effort estimates for all scenarios in the increment are summed to get an increment estimate

Estimation for WebApp Projects

- Modify the function point estimation procedure as follows
    a. *Inputs* are each input screen or form, each maintenance screen, and if you use a tab notebook metaphor each tab.
    b. *Outputs* are each static Web page, each dynamic web page script and each report
    c. *Tables* are each logical table in the database plus, if you are using XML to store data in a file, each XML object (or collection of XML attributes).
    d. *Interfaces* retain their definition as logical files into our out-of-the-system boundaries.
    e. *Queries* are each externally published or use a message-oriented interface.

Make-Buy Decision

- It may be more cost effective to acquire a piece of software rather than develop it.
- Decision tree analysis provides a systematic way to sort through the make-buy decision.
- As a rule outsourcing software development requires more skillful management than in-house development of the same product.

References :

1. Pressman R. S. , “Software Engineering – A practitioner’s approach”, Tata McGraw Hill.
2. Sommerville, “Software Engineering”, Pearson Education.
3. Pfleeger, “Software Engineering: Theory and Practice”, Pearson Education.

4. P. Jalote, “An Integrated approach to Software Engineering”, Narosa Publications.