Software Project Management

Lecture # 7

Outline

- Project Scheduling (Chapter 24)
  - What is Scheduling?
  - What is Tracking?
  - Project Scheduling
  - Late Software Delivery & Project Deadlines
  - Project Scheduling Evolution and Basic Scheduling Principles
  - People and Effort Relationship
  - Effort distribution
  - Defining task set for the software project
    - Project types (related topic)
  - Factors that influence task set selection in a project
  - A task set example
  - Defining a task network

Introduction

- When…
  - Appropriate process model has been selected,
  - Software engg. tasks have been identified,
  - Estimation of amount of work & people has been done,
  - Risk have been considered and deadline is known…
- Then its time to connect the dots…
  - to create a network of tasks for achieving the software engineering tasks. This network creation is called *‘software project scheduling’*

What is Project Scheduling?

- An activity that distributes estimated effort across the planned project duration by allocating the effort to specific software engineering task.
- Creating a network of software engineering tasks to complete the project and assign responsibilities of tasks and timing of tasks.

What is Project Tracking?

- Tracking is the process to make sure that all tasks are completed according to assigned responsibility and schedule.

Proper Project Scheduling

- Proper Project Scheduling requires:
  - All tasks should appear in the network,
  - Interdependencies between tasks are indicated,
  - Effort and timing are intelligently allocated to tasks,
  - Resources are allocated to tasks,
  - Closely spaced milestones are provided for progress tracking.

Who creates Project Schedule?

- At the project level, it is done by project managers using information solicited from software engineers.

Reasons for late software delivery

- Unrealistic deadline established by some one outside the software development group & enforced.
- Changing customer requirements that are not reflected in schedule change.
- An honest underestimate of the amount of work and/or resources required.
- Risks that were not considered at project commencement.

Reasons for late software delivery

- Technical difficulties not foreseen in advance.
- Miscommunication among project staff.
- A failure by project management to recognize that the project is falling behind schedule and a lack of action to correct the problem.

Dealing With Project Deadlines

- Aggressive (actually unrealistic) deadlines are a fact of life in software business.
- If best estimates indicate that deadline is unrealistic Project Manager should :

  "Protect his/her team from undue (schedule) pressure… and reflect pressure back to its originators."

  The above was stated in *Practical Project Management* by Page-Jones, M., *1985*

Dealing With Project Deadlines

- Recommended steps for such situations:
  1. Perform a detailed estimate using historical data from past projects. Determine estimated effort and duration for project.
  2. Use incremental model, develop a strategy that will deliver critical functionality within imposed deadline, but delay other functionality until later. Document the plan.
  3. Meet the customer and explain why imposed deadline is unrealistic. Explain what is the new time required to complete this project.

Dealing With Project Deadlines

3. Offer incremental development strategy as alternative. For example, offer some options:
   - First, we can increase the budget and have bring resources to get this job done in due time. But this contains increased risk of poor quality due to tight timeline.
   - Second, we can remove some software functions, and provide remaining functionality later.
   - Third, dispense with reality and wish to complete software in due time, say nine months (third option may be unacceptable).

- By presenting solid estimates and references to past projects, it is likely that, negotiated versions of option 1 and 2 will be accepted by customer.

Project Scheduling (Evolution)

- It is important to note that the project schedule evolves over time.
- During early stages of project planning, a macroscopic schedule is developed.
- This schedule identifies all major process framework activities and the product functions to which they are applied.
- As the project proceeds, each entry on the macroscopic schedule gets refined into detailed schedule.
- Specific tasks are identified to achieve each activity and are scheduled.

Project Scheduling - Basic Principles

- Compartmentalization
  - Both the product and the process are decomposed into a number of manageable activities/tasks
- Interdependency
  - Interdependencies among decomposed activities must be identified.
  - Some tasks can be performed in sequence and other can be done in parallel.
  - Some activities can not be performed without completion of another and some can be totally independent
- Time Allocation
  - Each task must be allocated work units (person-days of effort)
  - Start and end time must be allocated considering interdependencies & whether work will conducted be part-time or full-time

Project Scheduling - Basic Principles

- Effort validation
  - Project manager must ensure that no more than the allocated no. of people have been scheduled at any given time
- Defined responsibilities
  - Every scheduled task must be assigned to a specific team member
- Defined outcomes
  - Work products must be defined for every scheduled task
- Defined milestones
  - Every task/group of tasks must be associated with a project milestone. A milestone is accomplished after one or more related work products has been reviewed for quality and approved

Relationship between People and Effort

- Common Myth …
  - “If we fall behind schedule, we can always add more programmers and catch up later in the project!”
- Doing so is often disruptive rather than productive causing further delays. Reasons:
  - learning time for added people
  - teaching takes time away from productive work
  - Addition of people increases communication paths – hence increased complexity

Relationship between People and Effort

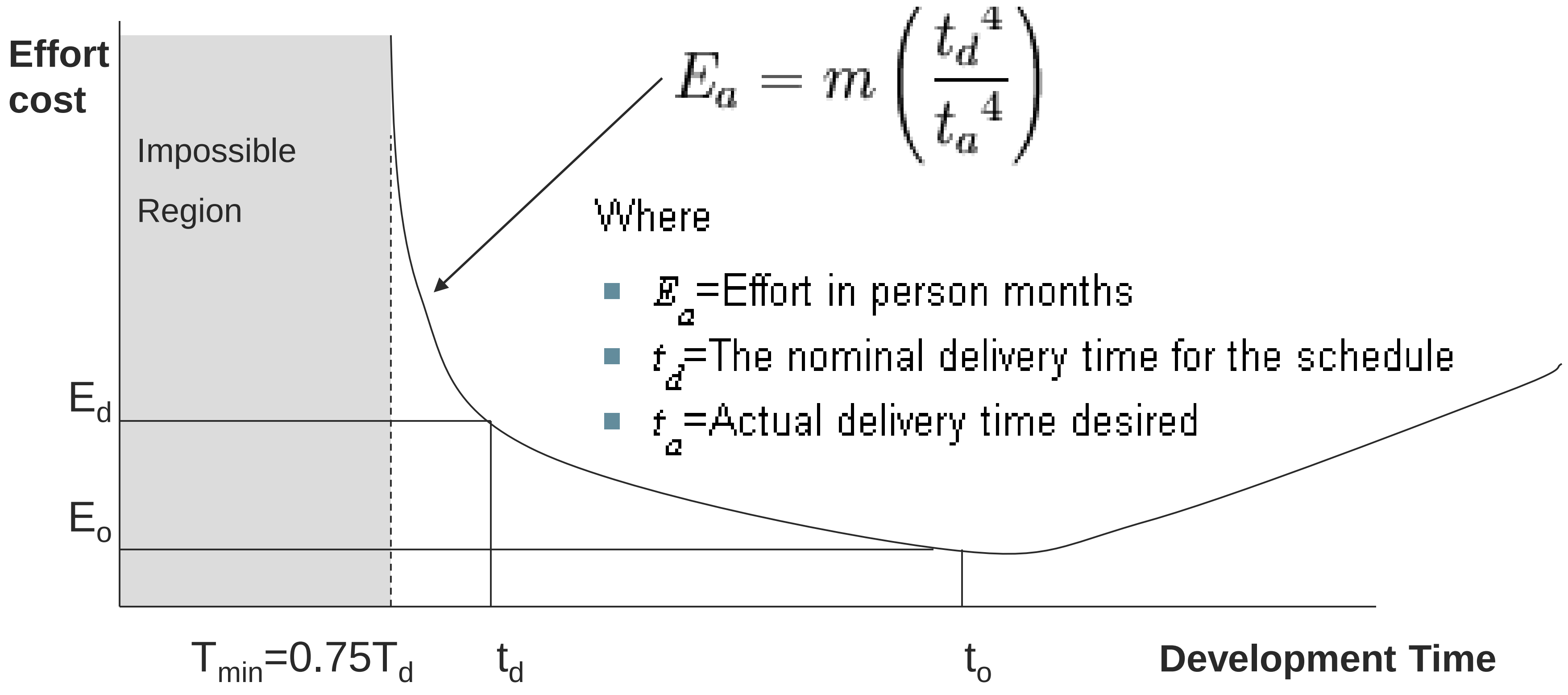
- Putnam-Norden-Rayleigh (PNR) Curve indicates the relationship between effort applied and delivery time for a software project.

$$E_a = m\left(\frac{t_d{}^4}{t_a{}^4}\right)$$

Where

- E_a=Effort in person months
- t_d=The nominal delivery time for the schedule
- t_a=Actual delivery time desired

- t_o = delivery time that will result in least effort expended
- As we move left to t_o, i.e. as we try to accelerate delivery, curve rises nonlinearly
- As we try to reduce accelerate delivery, curve rises sharply to left of t_d indicating, project delivery time can not be compressed much beyond $0.75t_d$
- As we try further, the project moves into impossible region and failure risk becomes high



$$E_a = m\left(\frac{{t_d}^4}{{t_a}^4}\right)$$

Where

- E_a=Effort in person months
- t_d=The nominal delivery time for the schedule
- t_a=Actual delivery time desired

PNR Curve & Software Eqn.

- The software equation is derived from the PNR curve
- It demonstrates a highly nonlinear relationship between time to complete project and human effort applied to the project
- Lines of Code (L) is related to effort (E) and development time (t) as:
- $L = P \times E^{1/3} t^{4/3}$
- Rearranging the equation, we get the expression for development effort
- $E = L^3 / P^3 t^4$

Effort Distribution

- A recommended distribution of effort across software process is often referred to as *40-20-40 rule*
  - *40%* allocated to analysis & design
  - *20%* allocated to coding
  - *40%* allocated to testing

- Use the above as a guideline only as each project dictates its own distribution effort, e.g. criticality of the software often dictates the amount of testing and hence testing effort

Defining Task Set For The Software Project

- Task set is a collection of software engineering work tasks , milestones and deliverables that must be accomplished to complete a particular software project.
- Task sets are different for different types of projects.

Defining Task Set For The Software Project (Contd.)

- Most organizations encounter following <u>types of projects</u>
  - *Concept development projects*
    - Explore some new business concept or application of new technology.
  - *New application development projects*
    - Undertaken as a consequence of specific customer request

Defining Task Set For The Software Project (Contd.)

- *Application Enhancement projects*
  - Involve modification to functions, performance or interfaces (observable by end-user) in existing software
- *Application maintenance projects*
  - That correct, adapt or extend existing software in ways that may not be obvious to end user
- *Reengineering projects*
  - Undertaken for rebuilding an existing system in whole or part

Factors Influencing Task Set Selection in Projects

- Size of project
- Number of potential users
- Mission criticality
- Application longevity
- Stability of requirements
- Ease of customer/develop communication
- Maturity of applicable technology
- Performance constraints
- Embedded, non embedded characteristics
- Project staff
- Reengineering factors
- *These factors also provide an indication of the degree of rigor with which the software process should be applied*

A Task Set Example

- Consider software engineering tasks for a <u>Concept Development</u> project.
- Such projects are approached by applying the following major tasks:

  1.1 Concept scoping
    - determines overall project scope

  1.2 Preliminary concept planning
    - Establishes the organization’s ability to undertake work implied by project scope

  1.3 Technology risk assessment
    - Evaluates risk associated with the technology to be implemented

A Task Set Example (Contd.)

1.4 Proof of concept

- Demonstrates the viability of a new technology in the software context

1.5 Concept implementation

- Implements the concept representation in a manner that can be reviewed by a customer and is used for marketing purposes when a concept must be sold to other customers or management.

1.6 Customer Reaction

- Concept solicits feedback on a new technology concept and targets specific customer applications

Refinement of Major Tasks

- The major tasks described earlier may be used to define a macroscopic schedule for project.
- The macroscopic schedule must be refined to create a detailed schedule.
- For this each major task is decomposed into a set of subtasks (with related work products and milestones)
- As an example consider task 1.1 – concept scoping. The refinement is shown on next slide:

Refinement of Major Tasks (Contd.)

Task definition: Task 1.1 Concept Scoping

- 1.1.1 identify need, benefits and potential customers
- 1.1.2 define desired output/control and input events that drive the application.
  - Begin task 1.1.2
    - 1.1.2.1 FTR: review ‘written description’ of need
    - 1.1.2.2 Derive a list of customer visible o/p, i/p
    - 1.1.2.3 FTR: Review o/p, i/p with customers and revise as required
  - End task 1.1.2
- 1.1.3 Define the functionality/behaviour of each major function
  - Begin task 1.1.3
    - 1.1.3.1 FTR: Review o/p, i/p data objects derived in task 1.1.2
    - 1.1.3.2 Derive a model of functions/behavior
    - 1.1.3.3 FTR: Review functions/behavior with customers and revise as required
  - End task 1.1.3

Refinement of Major Tasks (Contd.)

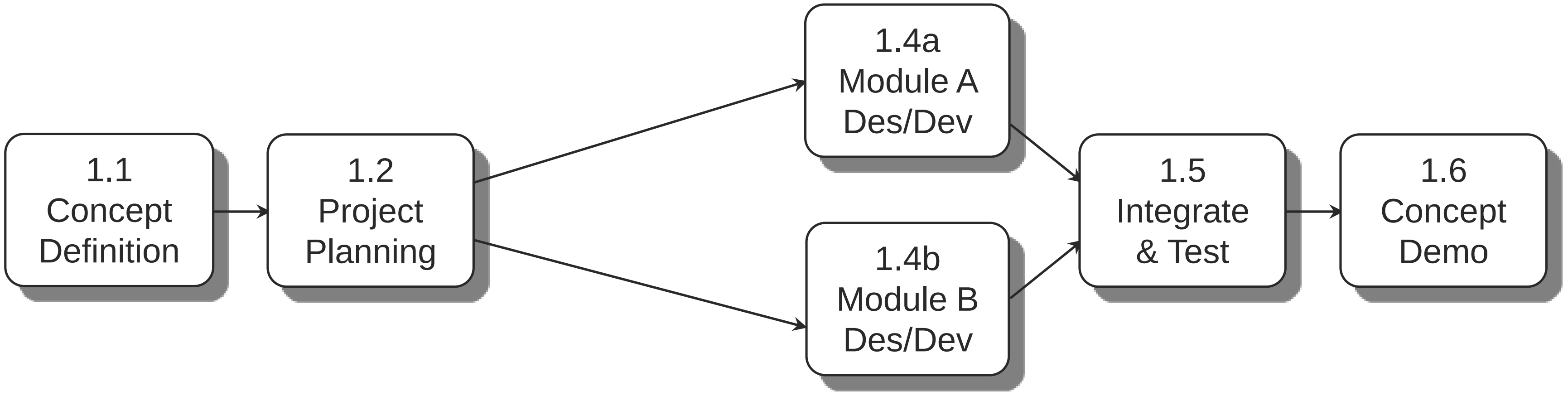
- 1.1.4 Isolate those elements of technology to be implemented in software
- 1.1.5 Research availability of existing software
- 1.1.6 Define technical feasibility
- 1.1.7 make quick estimate of size
- 1.1.8 Create a scope Definition

End task definition: Task 1.1

Defining a Task Network

- Also known as *activity network.*
- It is a graphic representation of the task flow for a project.
- Displays interdependencies and parallelism.
- Project manager should be aware of those tasks that lie on the critical path.

A task set network for 'concept development'



Enables the team to see the essentially serial nature of the project, but take advantage of parallelism where possible.

References

- Today’s lecture contents were taken from
  - Chapter 24 from “Software Engineering, A Practitioner's Approach” by Roger Pressman