Introduction to Software Engineering

(CS350)

Lecture 04

Jongmoon Baik

Project Mgmt. & Scheduling

The Four P's

- People — the most important element of a successful project
- Product — the software to be built
- Process — the set of framework activities and software engineering tasks to get the job done
- Project — all work required to make the product a reality

Stakeholders

- ***Senior managers*** who define the business issues that often have significant influence on the project.
- ***Project (technical) managers*** who must plan, motivate, organize, and control the practitioners who do software work.
- ***Practitioners*** who deliver the technical skills that are necessary to engineer a product or application.
- ***Customers*** who specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome.
- ***End-users*** who interact with the software once it is released for production use.

Software Teams

How to lead?

How to organize?

How to collaborate?

How to motivate?

How to create good ideas?

Team Leader

- The MOI Model of leadership
  - **Motivation.** The ability to encourage (by "push or pull") technical people to produce to their best ability.
  - **Organization.** The ability to mold existing processes (or invent new ones) that will enable the initial concept to be translated into a final product.
  - **Ideas or innovation.** The ability to encourage people to create and feel creative even when they must work within bounds established for a particular software product or application.

Jerry Weinberg, 1986

Software Teams

The following factors must be considered when selecting a software project team structure ...

- Difficulty of the problem to be solved
- Size of the resultant program(s) in lines of code or function points
- Time that the team will stay together (team lifetime)
- Degree to which the problem can be modularized
- Required quality and reliability of the system to be built
- Rigidity of the delivery date
- Degree of sociability (communication) required for the project

Organizational Paradigms

1. Closed paradigm—structures a team along a traditional hierarchy of authority
2. Random paradigm—structures a team loosely and depends on individual initiative of the team members
3. Open paradigm—attempts to structure a team in a manner that achieves some of the controls associated with the closed paradigm but also much of the innovation that occurs when using the random paradigm
4. Synchronous paradigm—relies on the natural compartmentalization of a problem and organizes team members to work on pieces of the problem with little active communication among themselves

suggested by Constantine [Con93]

Avoid Team “Toxicity”

- A frenzied work atmosphere in which team members waste energy and lose focus on the objectives of the work to be performed.
- High frustration caused by personal, business, or technological factors that cause friction among team members.
- “Fragmented or poorly coordinated procedures” or a poorly defined or improperly chosen process model that becomes a roadblock to accomplishment.
- Unclear definition of roles resulting in a lack of accountability and resultant finger-pointing.
- “Continuous and repeated exposure to failure” that leads to a loss of confidence and a lowering of morale.

Agile Teams

- To perform a high-performance team
  - Team members must have trust in one another.
  - The distribution of skills must be appropriate to the problem.
  - Mavericks may have to be excluded from the team, if team cohesiveness is to be maintained.
- Team is “self-organizing”
  - An adaptive team structure
  - Uses elements of Constantine’s random, open, and synchronous paradigms
  - Significant autonomy
    - allowed to select its own approach (e.g. process, methods, tools)

Team Coordination & Communication

- *Formal, impersonal approaches* include software engineering documents and work products (including source code), technical memos, project milestones, schedules, and project control tools (Chapter 23), change requests and related documentation, error tracking reports, and repository data (see Chapter 26).
- *Formal, interpersonal procedures* focus on quality assurance activities (Chapter 25) applied to software engineering work products. These include status review meetings and design and code inspections.
- *Informal, interpersonal procedures* include group meetings for information dissemination and problem solving and "collocation of requirements and development staff."
- *Electronic communication* encompasses electronic mail, electronic bulletin boards, and by extension, video-based conferencing systems.
- *Interpersonal networking* includes informal discussions with team members and those outside the project who may have experience or insight that can assist team members.

The Product Scope

- Scope
    - **Context.** How does the software to be built fit into a larger system, product, or business context and what constraints are imposed as a result of the context?
    - **Information objectives.** What customer-visible data objects (Chapter 8) are produced as output from the software? What data objects are required for input?
    - **Function and performance.** What function does the software perform to transform input data into output? Are any special performance characteristics to be addressed?
- Software project scope must be unambiguous and understandable at the management and technical levels.

Problem Decomposition

- Sometimes called *partitioning* or *problem elaboration*
- Once scope is defined …
  - It is decomposed into constituent functions
  - It is decomposed into user-visible data objects

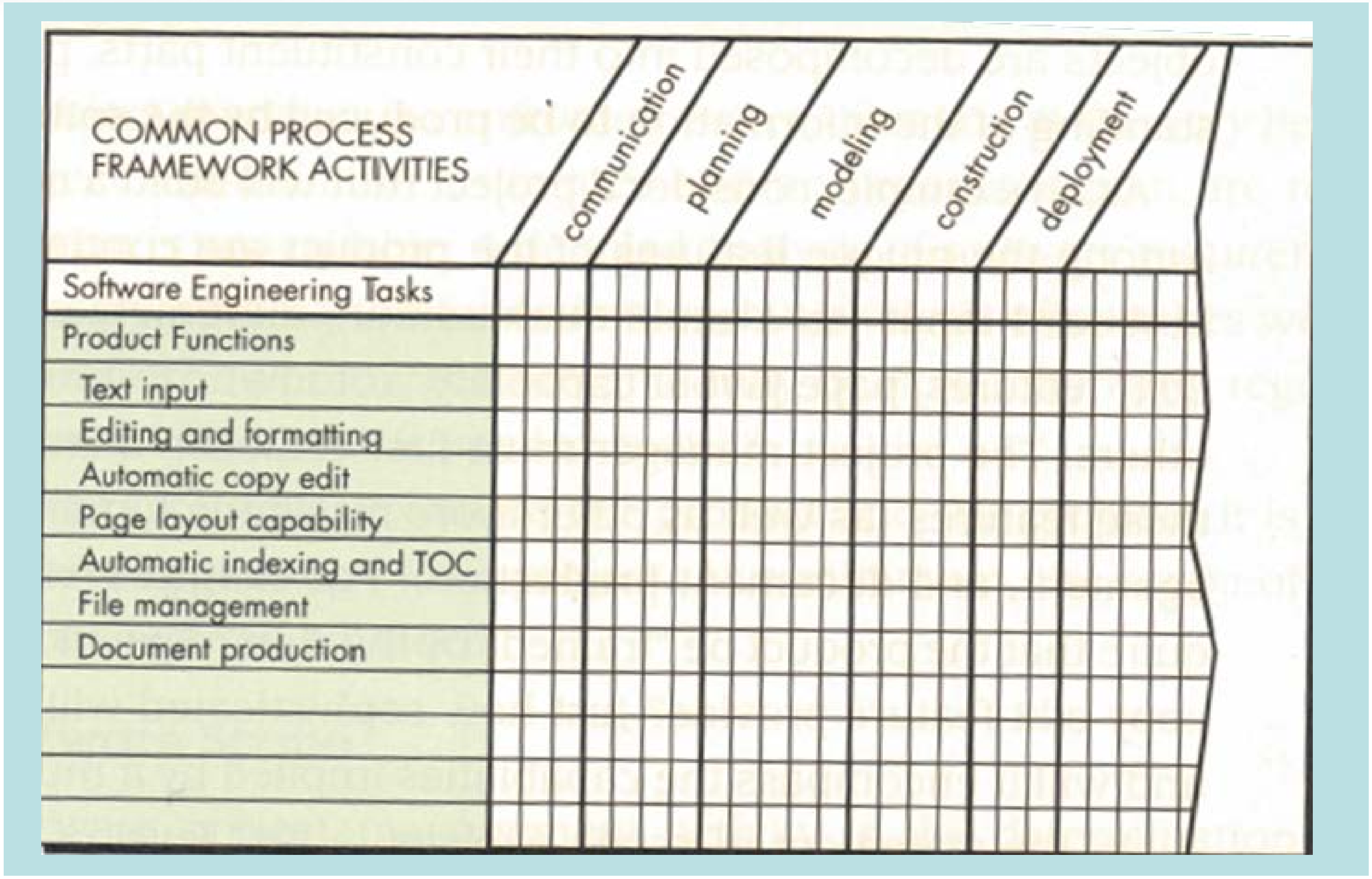
  or

  - It is decomposed into a set of problem classes
- Decomposition process continues until all functions or problem classes have been defined

The Process

- Once a process framework has been established
  - Consider project characteristics
  - Determine the degree of rigor required
  - Define a task set for each software engineering activity
    - Task set =
      - Software engineering tasks
      - Work products
      - Quality assurance points
      - Milestones

Melding the Problem and the Process



COMMON PROCESS FRAMEWORK ACTIVITIES	communication	planning	modeling	construction	deployment
Software Engineering Tasks					
Product Functions					
Text input					
Editing and formatting					
Automatic copy edit					
Page layout capability					
Automatic indexing and TOC					
File management					
Document production					

The Project

- *Projects get into trouble when …*
  - Software people don’t understand their customer’s needs.
  - The product scope is poorly defined.
  - Changes are managed poorly.
  - The chosen technology changes.
  - Business needs change [or are ill-defined].
  - Deadlines are unrealistic.
  - Users are resistant.
  - Sponsorship is lost [or was never properly obtained].
  - The project team lacks people with appropriate skills.
  - Managers [and practitioners] avoid best practices and lessons learned.

Common-Sense Approach to Projects

1. *Start on the right foot.* This is accomplished by working hard (very hard) to understand the problem that is to be solved and then setting realistic objectives and expectations.
2. *Maintain momentum. The* project manager must provide incentives to keep turnover of personnel to an absolute minimum, the team should emphasize quality in every task it performs, and senior management should do everything possible to stay out of the team's way.
3. *Track progress.* For a software project, progress is tracked as work products (e.g., models, source code, sets of test cases) are produced and approved (using formal technical reviews) as part of a quality assurance activity.
4. *Make smart decisions.* In essence, the decisions of the project manager and the software team should be to "keep it simple."
5. *Conduct a postmortem analysis.* Establish a consistent mechanism for extracting lessons learned for each project.

To Get to the Essence of a Project

- Why is the system being developed?
- What will be done?
- When will it be done?
- Who is responsible for a function?
- Where are they located organizationally?
- How will the job be done technically and managerially?
- How much of each resource (e.g., people, software, tools, database) will be needed?

Barry Boehm [Boe96]

Critical Practices

- Formal risk management
- Empirical cost and schedule estimation
- Metrics-based project management
- Earned value tracking
- Defect tracking against quality targets
- People aware project management

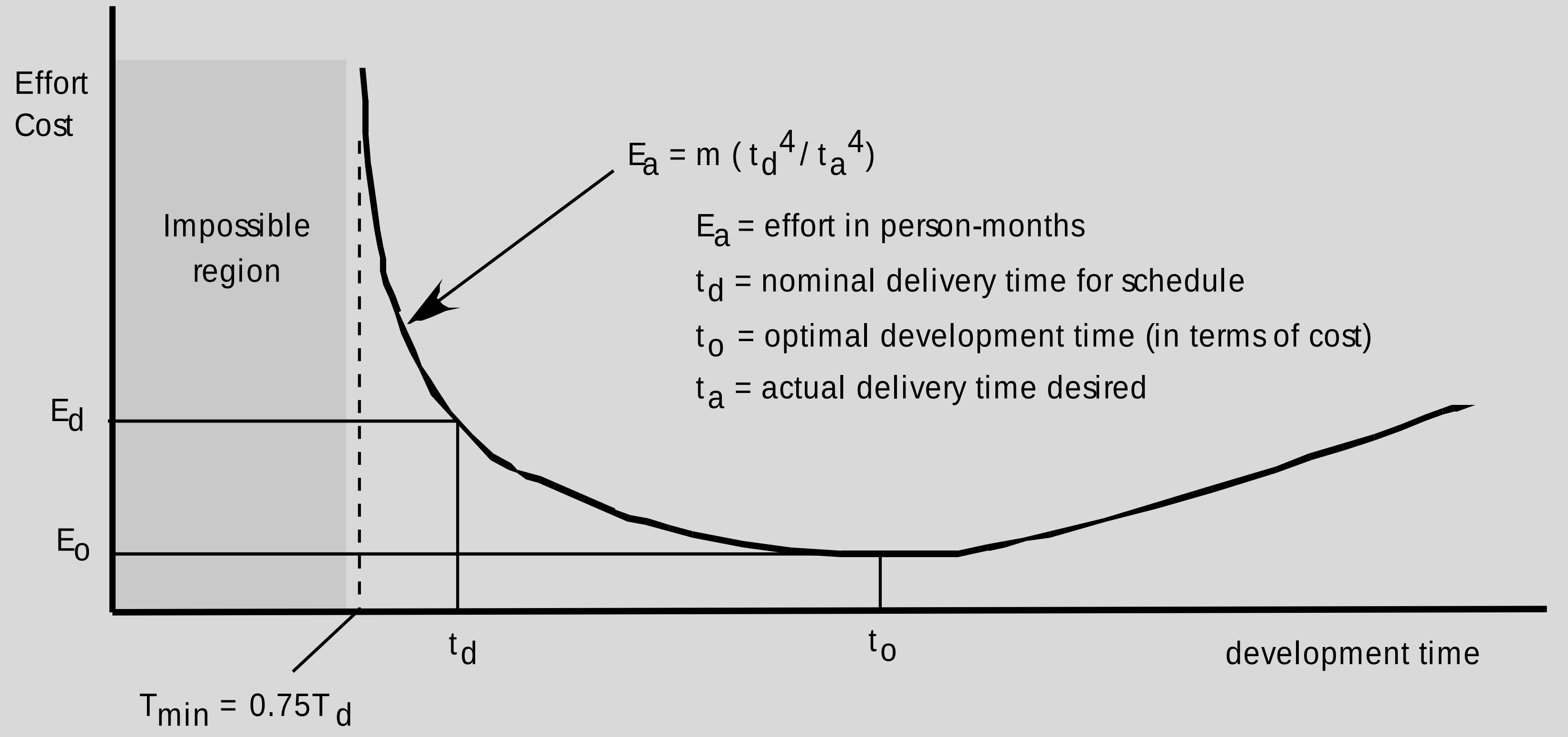
Why Are Projects Late?

- an unrealistic deadline established by someone outside the software development group
- changing customer requirements that are not reflected in schedule changes;
- an honest underestimate of the amount of effort and/or the number of resources that will be required to do the job;
- predictable and/or unpredictable risks that were not considered when the project commenced;
- technical difficulties that could not have been foreseen in advance;
- human difficulties that could not have been foreseen in advance;
- miscommunication among project staff that results in delays;
- a failure by project management to recognize that the project is falling behind schedule and a lack of action to correct the problem

Scheduling Principles

- **Compartmentalization :** Partition into a number of manageable activities and tasks; Refine both process and product
- **Interdependency:** Determine the inter-relationships among the compartmentalized tasks and activities
- **Time allocation:** Allocate some number of work units to each task; Assign a start and a completion time
- **Effort validation:** Ensure that no more than the allocated number of people has been scheduled as any given time
- **Defined responsibilities:** Assign every task to a specific team member
- **Defined outcomes:** Define an outcome (work product) for every task
- **Defined milestones:** Review work products for quality and then approved

Effort and Delivery Time



$E_a = m\ (t_d^4 / t_a^4)$

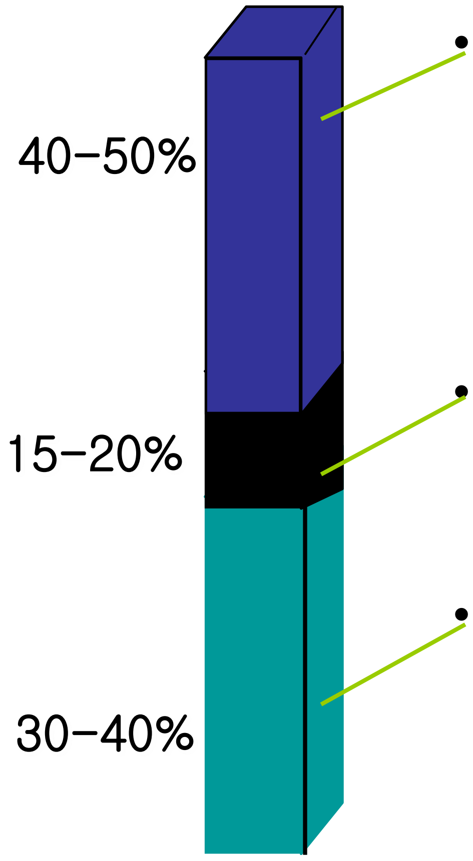
E_a = effort in person-months

t_d = nominal delivery time for schedule

t_o = optimal development time (in terms of cost)

t_a = actual delivery time desired

Effort Allocation



- “front end” activities
  - customer communication
  - analysis
  - design
  - review and modification
- construction activities
  - coding or code generation
- testing and installation
  - unit, integration
  - white-box, black box
  - regression

Defining Task Sets

- Determine type of project
- Assess the degree of rigor required
- Identify adaptation criteria
- Select appropriate software engineering tasks

Task Set Refinement

1.1 Concept scoping determines the overall scope of the project.

is refined to

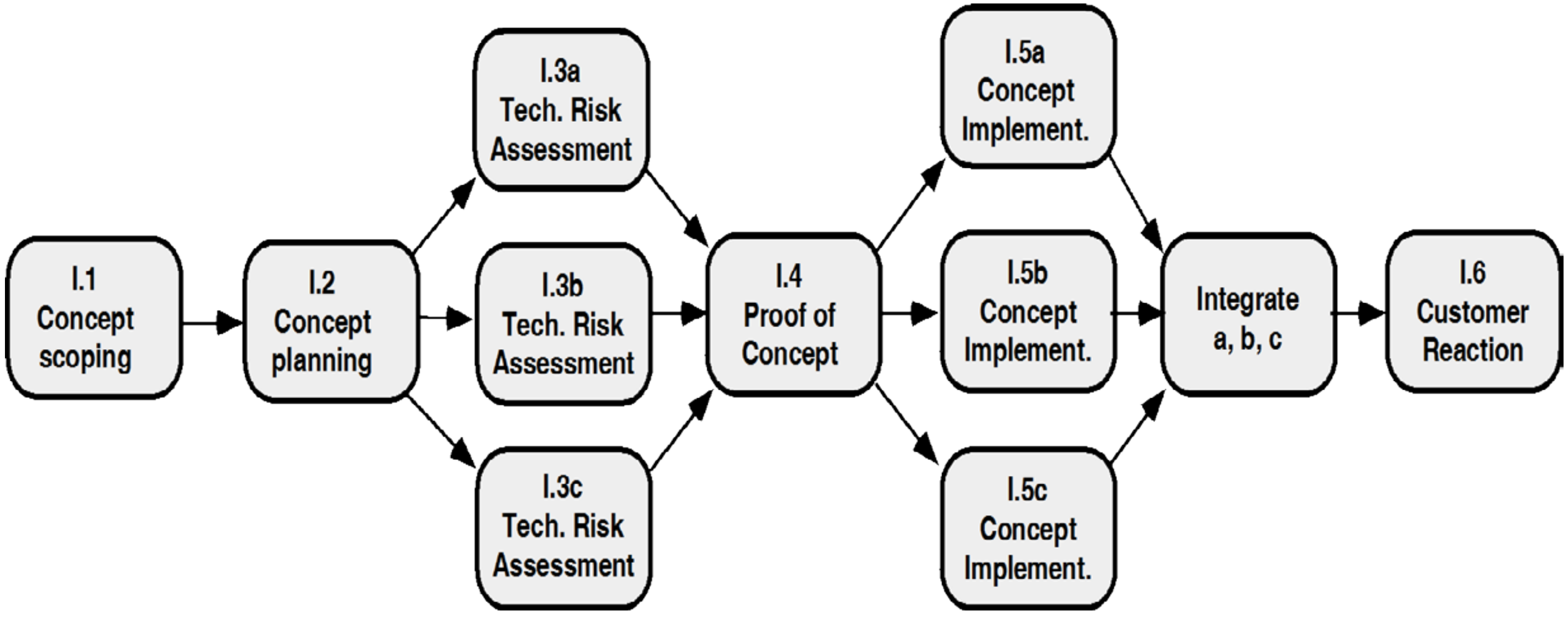
```
Task definition:  Task 1.1  Concept Scoping
1.1.1           Identify need, benefits and potential customers;
1.1.2           Define desired output/control and input events that drive the application;
                Begin Task 1.1.2
                1.1.2.1         FTR:  Review written description of need
```

FTR indicates that a formal technical review (Chapter 26) is to be conducted.

```
                1.1.2.2         Derive a list of customer visible outputs/inputs
                1.1.2.3         FTR:  Review outputs/inputs with customer and revise as required;
                endtask Task 1.1.2
1.1.3           Define the functionality/behavior for each major function;
                Begin Task 1.1.3
                1.1.3.1         FTR:  Review output and input data objects derived in task 1.1.2;
                1.1.3.2         Derive a model of functions/behaviors;
                1.1.3.3         FTR:  Review functions/behaviors with customer and revise as
required;
                endtask Task 1.1.3
1.1.4           Isolate those elements of the 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;
endTask definition:   Task 1.1
```

Define a Task Network

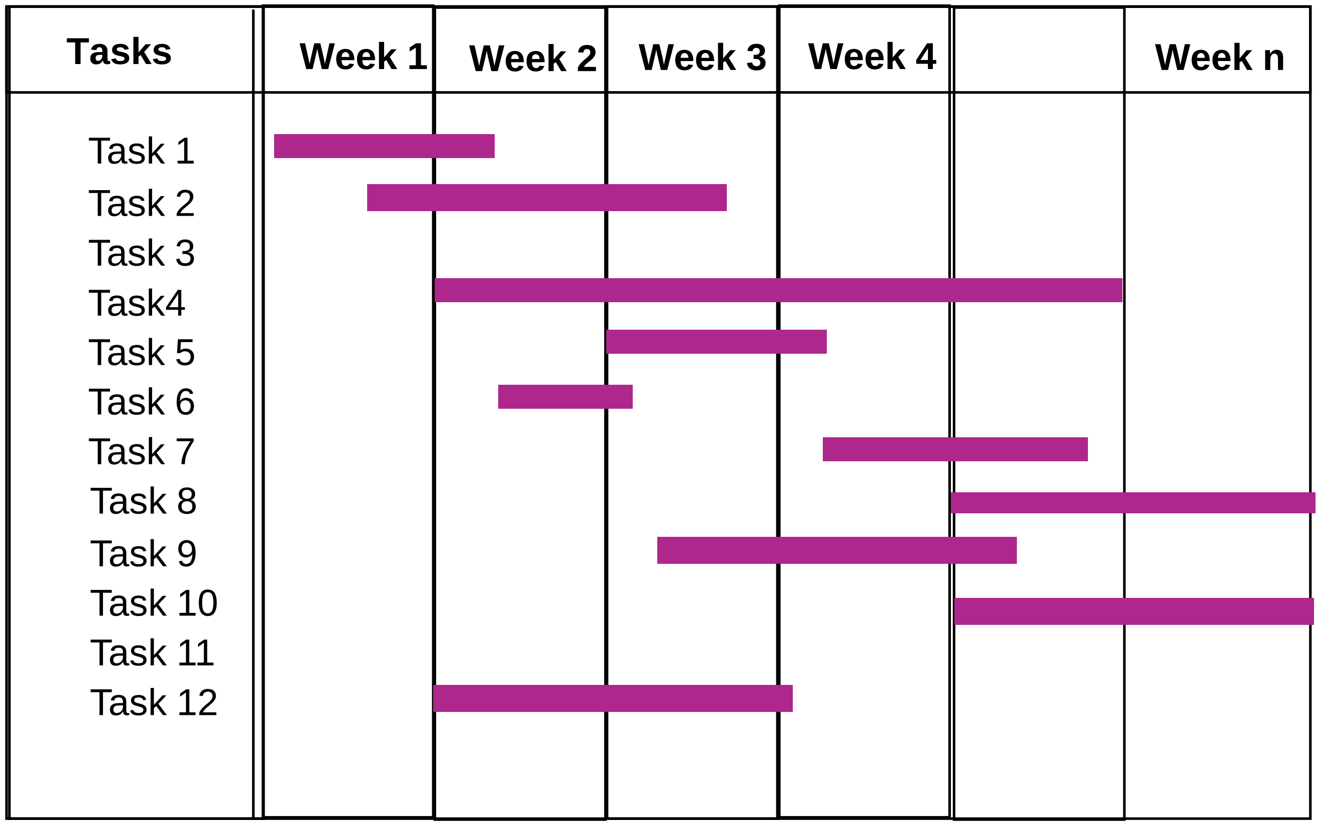
Activity Network or PERT



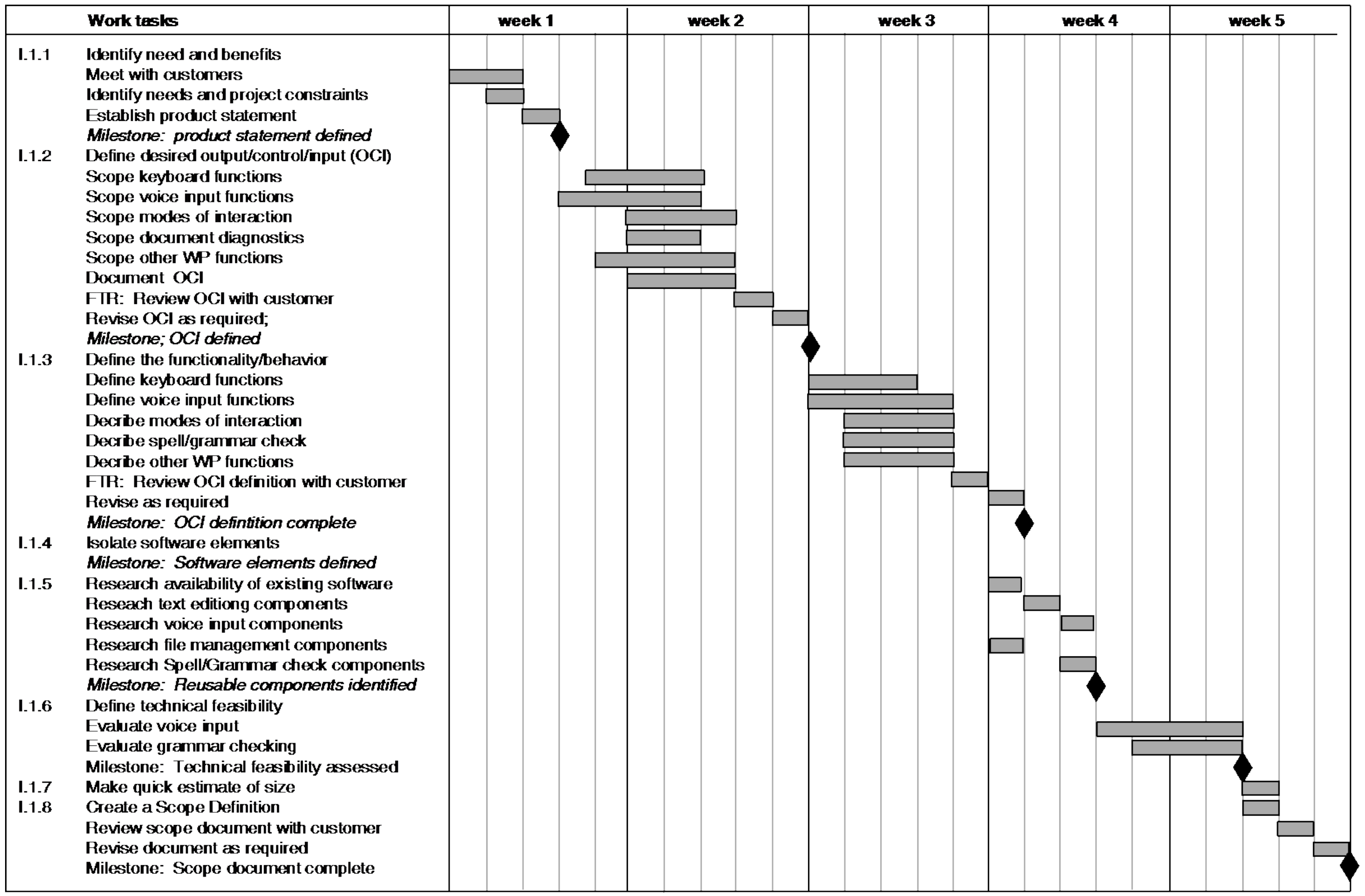
Three I.3 tasks are applied in parallel to 3 different concept functions

Three I.3 tasks are applied in parallel to 3 different concept functions

Timeline Charts (Gantt Chart)



Use Automated Tools to Derive a Timeline Chart



	Work tasks
I.1.1	Identify need and benefits
	Meet with customers
	Identify needs and project constraints
	Establish product statement
	Milestone: product statement defined
I.1.2	Define desired output/control/input (OCI)
	Scope keyboard functions
	Scope voice input functions
	Scope modes of interaction
	Scope document diagnostics
	Scope other WP functions
	Document OCI
	FTR: Review OCI with customer
	Revise OCI as required;
	Milestone; OCI defined
I.1.3	Define the functionality/behavior
	Define keyboard functions
	Define voice input functions
	Decribe modes of interaction
	Decribe spell/grammar check
	Decribe other WP functions
	FTR: Review OCI definition with customer
	Revise as required
	Milestone: OCI defintition complete
I.1.4	Isolate software elements
	Milestone: Software elements defined
I.1.5	Research availability of existing software
	Reseach text editiong components
	Research voice input components
	Research file management components
	Research Spell/Grammar check components
	Milestone: Reusable components identified
I.1.6	Define technical feasibility
	Evaluate voice input
	Evaluate grammar checking
	Milestone: Technical feasibility assessed
I.1.7	Make quick estimate of size
I.1.8	Create a Scope Definition
	Review scope document with customer
	Revise document as required
	Milestone: Scope document complete

Schedule Tracking

- Conduct periodic project status meetings in which each team member reports progress and problems.
- Evaluate the results of all reviews conducted throughout the software engineering process.
- Determine whether formal project milestones have been accomplished by the scheduled date.
- Compare actual start-date to planned start-date for each project task listed in the resource table.
- Meet informally with practitioners to obtain their subjective assessment of progress to date and problems on the horizon.
- Use earned value analysis to assess progress quantitatively.

Earned Value Analysis (EVA)

- Earned value
  - is a measure of progress
  - enables us to assess the “percent of completeness” of a project using quantitative analysis rather than rely on a gut feeling
  - “provides accurate and reliable readings of performance from as early as 15 percent into the project.” [Fle98]

Computing Earned Value-I

- The *budgeted cost of work scheduled* (BCWS) is determined for each work task represented in the schedule.
  - $BCWS_i$ is the effort planned for work task *i*.
  - To determine progress at a given point along the project schedule, the value of BCWS is the sum of the $BCWS_i$ values for all work tasks that should have been completed by that point in time on the project schedule.
- The BCWS values for all work tasks are summed to derive the *budget at completion,* BAC. Hence,

$$BAC = \sum (BCWS_k) \text{ for all tasks } k$$

Computing Earned Value-II

- Next, the value for *budgeted cost of work performed* (BCWP) is computed.
  – The value for BCWP is the sum of the BCWS values for all work tasks that have actually been completed by a point in time on the project schedule.
- “the distinction between the BCWS and the BCWP is that the former represents the budget of the activities that were planned to be completed and the latter represents the budget of the activities that actually were completed.” [Wil99]
- Given values for BCWS, BAC, and BCWP, important progress indicators can be computed:
  - Schedule performance index, SPI = BCWP/BCWS
  - Schedule variance, SV = BCWP – BCWS
  - SPI is an indication of the efficiency with which the project is utilizing scheduled resources.

Computing Earned Value-III

- Percent scheduled for completion = BCWS/BAC
  - provides an indication of the percentage of work that should have been completed by time *t*.
- Percent complete = BCWP/BAC
  - provides a quantitative indication of the percent of completeness of the project at a given point in time, *t*.
- *Actual cost of work performed,* ACWP, is the sum of the effort actually expended on work tasks that have been completed by a point in time on the project schedule. It is then possible to compute
  - Cost performance index, CPI = BCWP/ACWP
  - Cost variance, CV = BCWP – ACWP

Q & A