Chapter 2 – Software Processes

Topics covered

- ✧ Definitions
- ✧ Software process models (SDLS)
- ✧ Project Scheduling

Definition: The software process

- A **structured** set of activities required to develop a software system.
- A **software process model** is an abstract representation of a process. It presents **a description of a process** from some particular perspective.

Definition: Software process descriptions

- When we describe and discuss processes, we usually talk about
  - the activities in these processes such as specifying a data model, designing a user interface, etc.
  - and the ordering of these activities.
- Process descriptions may also include:
  - Products, which are the outcomes of a process activity;
  - Roles, which reflect the responsibilities of the people involved in the process;
  - Pre- and post-conditions, which are statements that are true before and after a process activity has been enacted or a product produced.

Definition: Plan-driven Vs. agile processes

- Plan-driven processes are processes where all of the process activities are planned in advance and progress is measured against this plan.
- In agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.
- In practice, most practical processes include elements of both plan-driven and agile approaches.
- There are no right or wrong software processes.

The software process

✧ Many different software processes but all involve:

- **Specification** – defining what the system should do;
- **Design and implementation** – defining the organization of the system and implementing the system;
- **Validation** – checking that it does what the customer wants;
- **Evolution** – changing the system in response to changing customer needs.

Quick Quiz

- What are the 4 different software processes in all SDLCs?

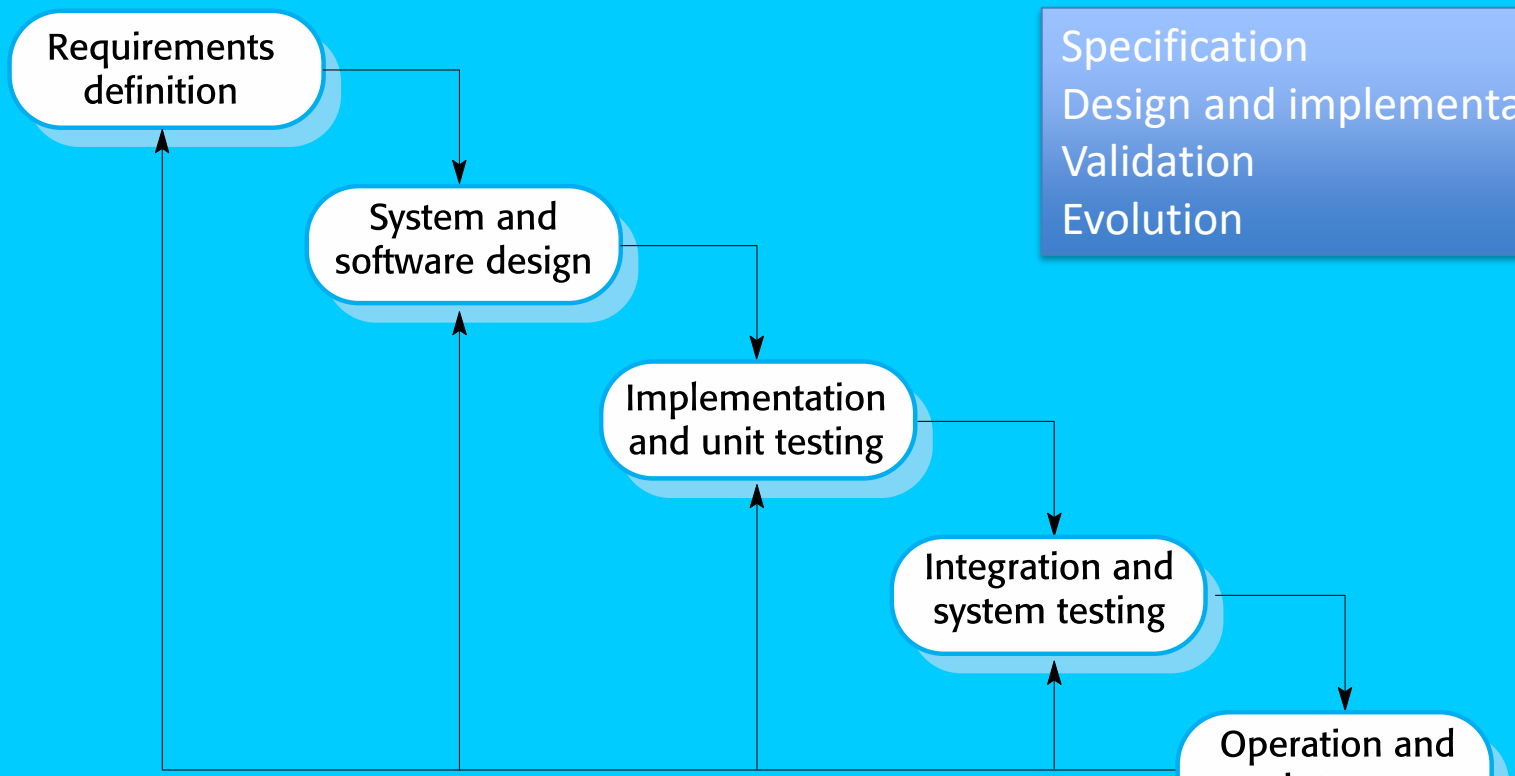
Specification
Design and implementation
Validation
Evolution

Software process models

Software process models

- The waterfall model
  - Plan-driven model. Separate and distinct phases of specification and development.
- Incremental development
  - Specification, development and validation are interleaved. May be plan-driven or agile.
- Integration and configuration
  - The system is assembled from existing configurable components. May be plan-driven or agile.
- In practice, most large systems are developed using a process that incorporates elements from all of these models.

The waterfall model

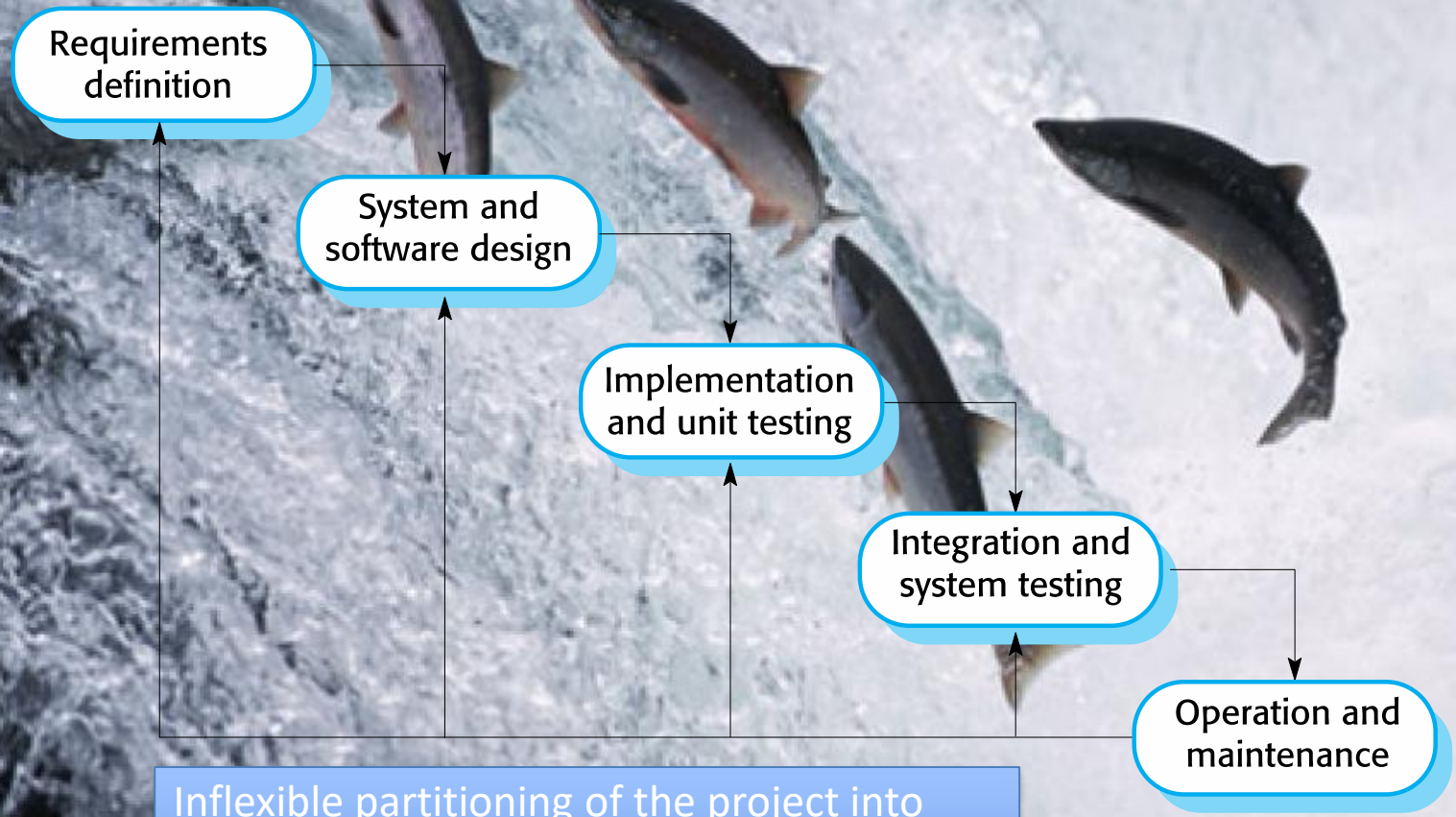


Specification
Design and implementation
Validation
Evolution

In principle, a phase has to be complete before moving onto the next phase.

The waterfall model

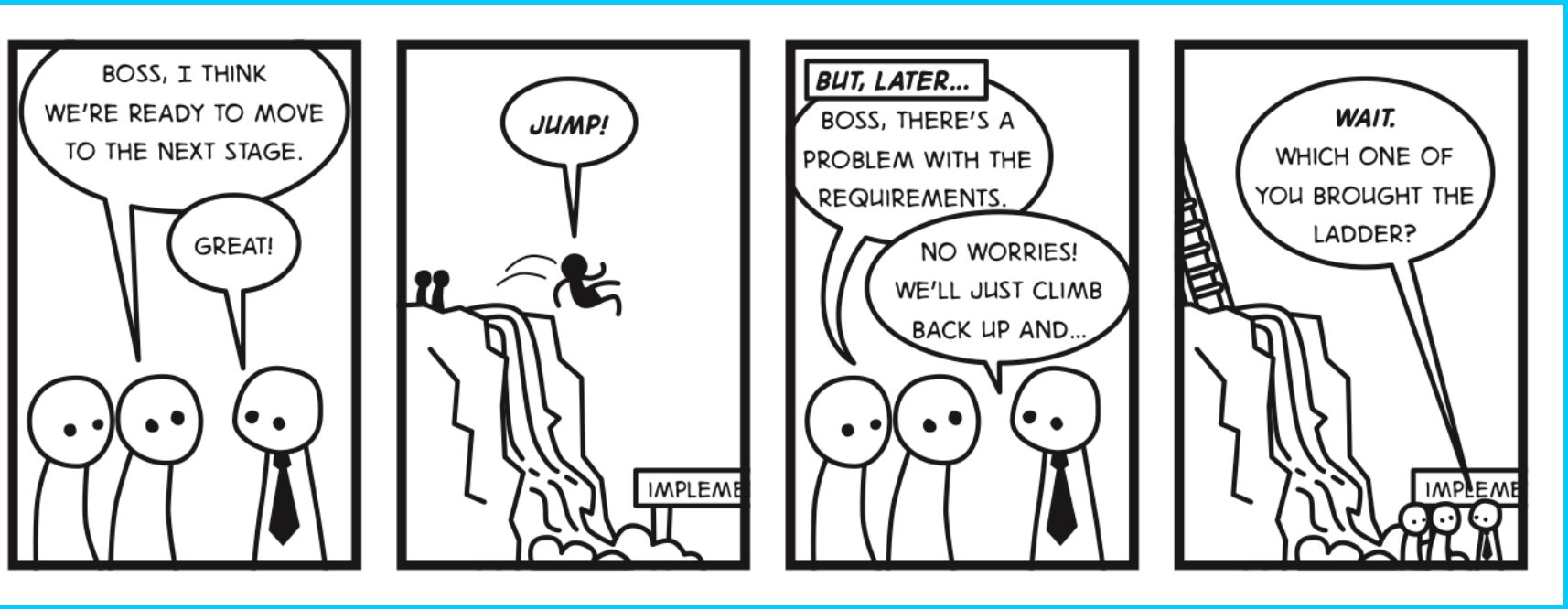
The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway.



Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.

Anything but Waterfall … but

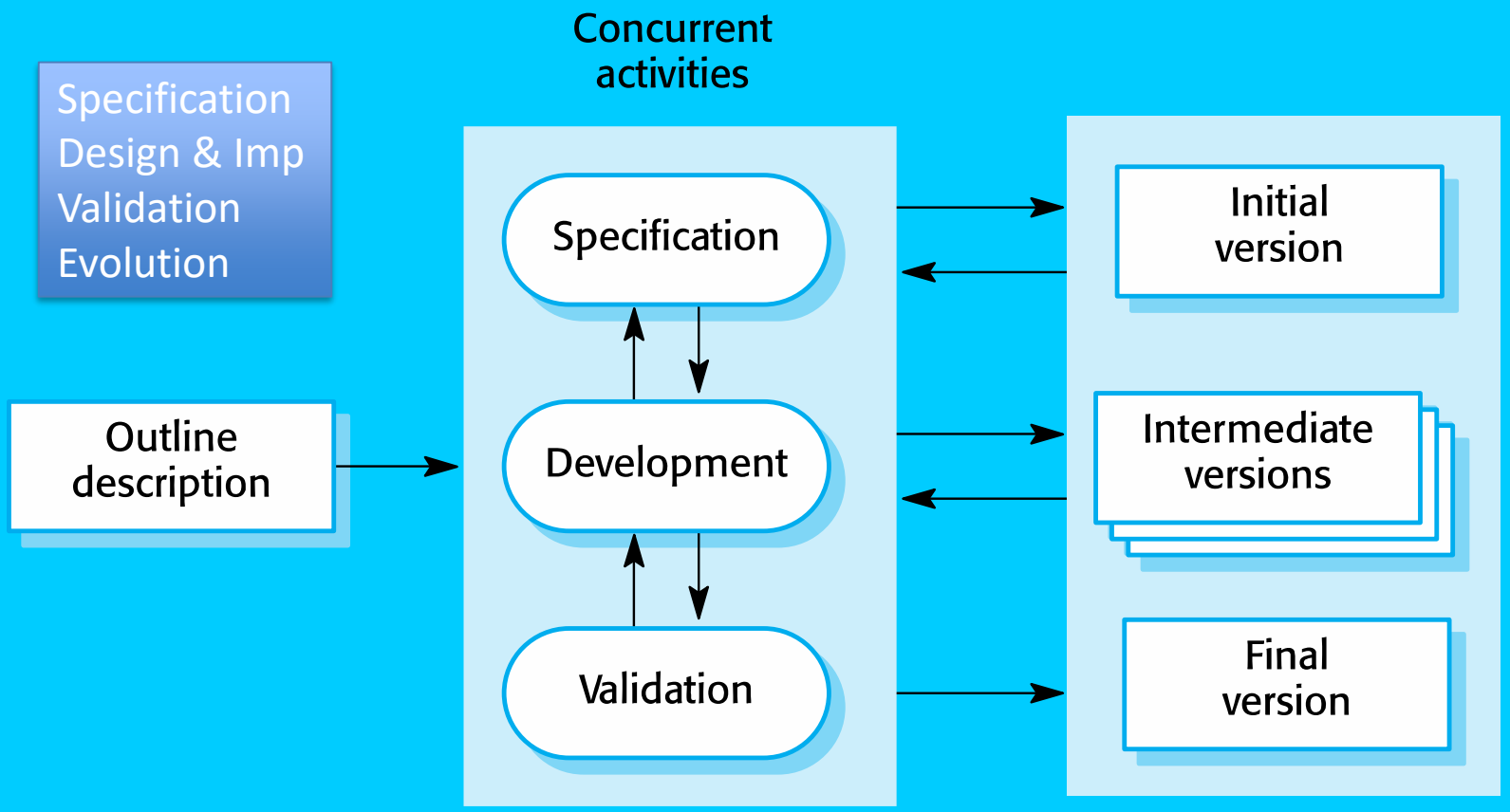
- The waterfall model was originally defined by Winston W. Royce in 1970.
  - Pong was not yet invented.
  - It was another 10 years before “A Rational Design Process. How and Why to Fake It”.



Use the Waterfall model when:

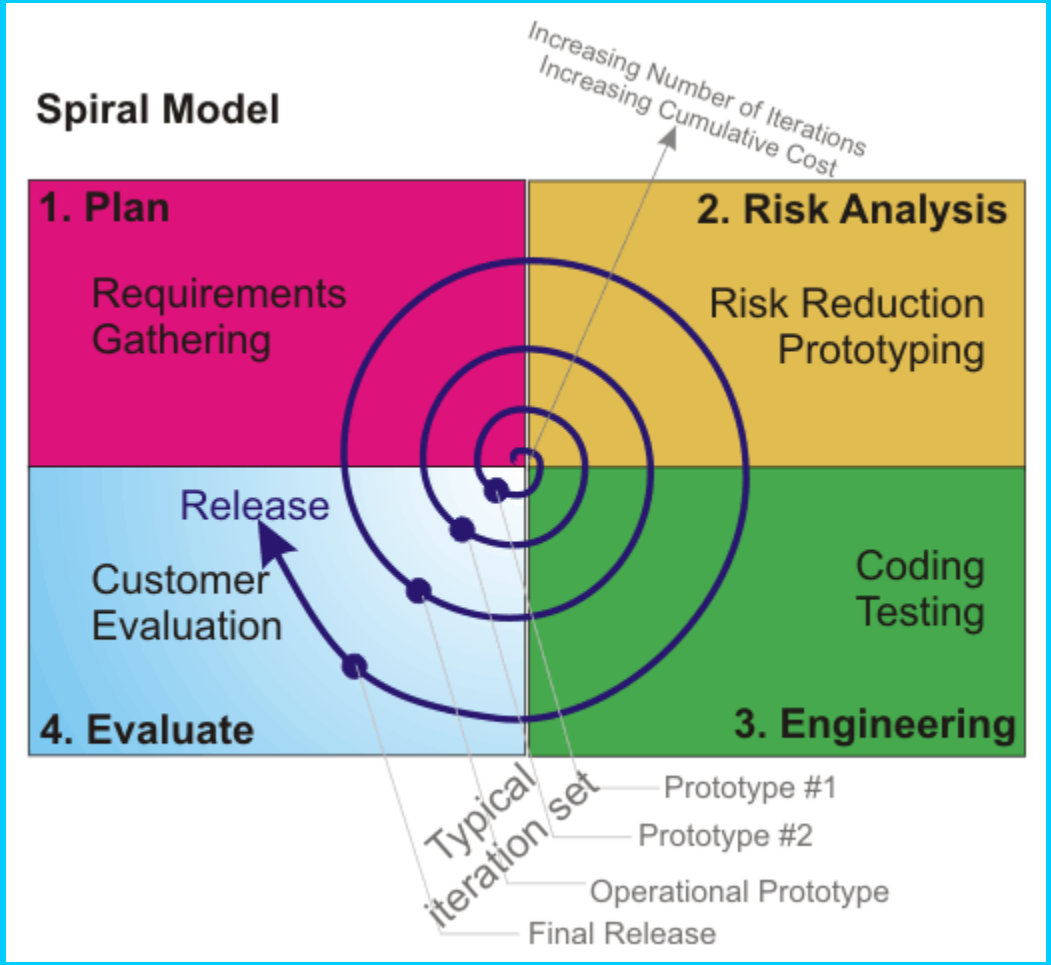
- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.
  - This model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
  - Few business systems have stable requirements but in those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

Incremental development



Incremental development is closely related to the Spiral model

- Both are meta-models (they are used as elements of other models).
- The primary difference is in how you decide what features come first.
  - Incremental takes a vertical slice of user requirements
  - Spiral minimizes risk

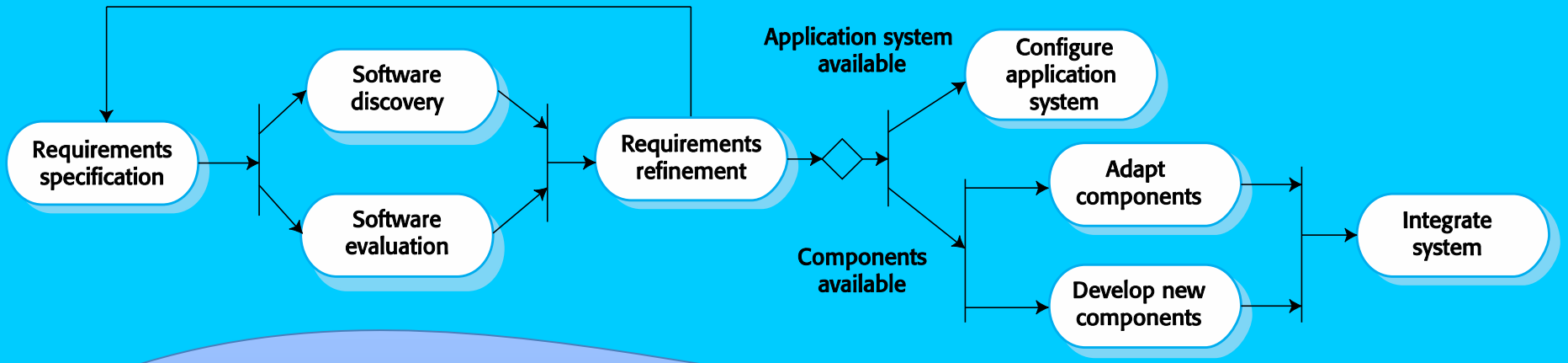


Use the Spiral model when:

- Almost every large project uses this meta-model in some form or another.

Agile uses this meta-model

Reuse-oriented software engineering

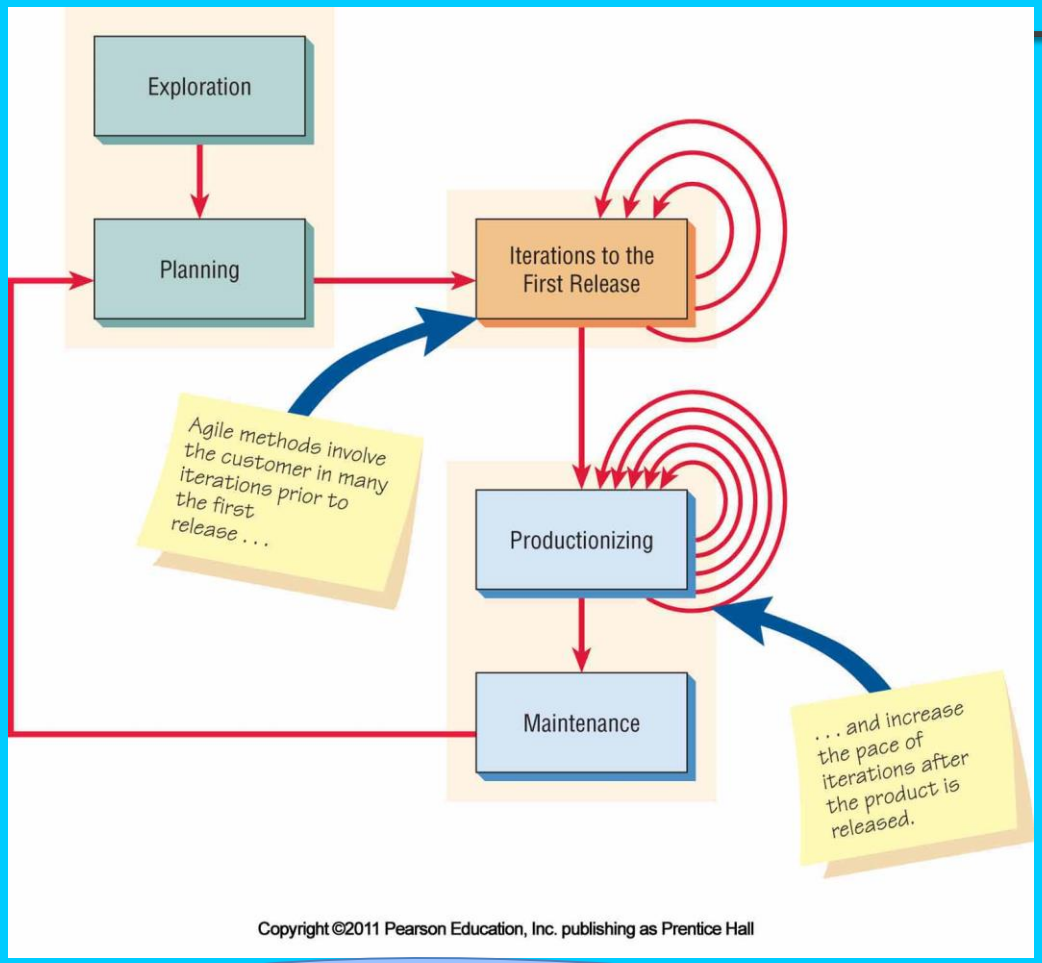


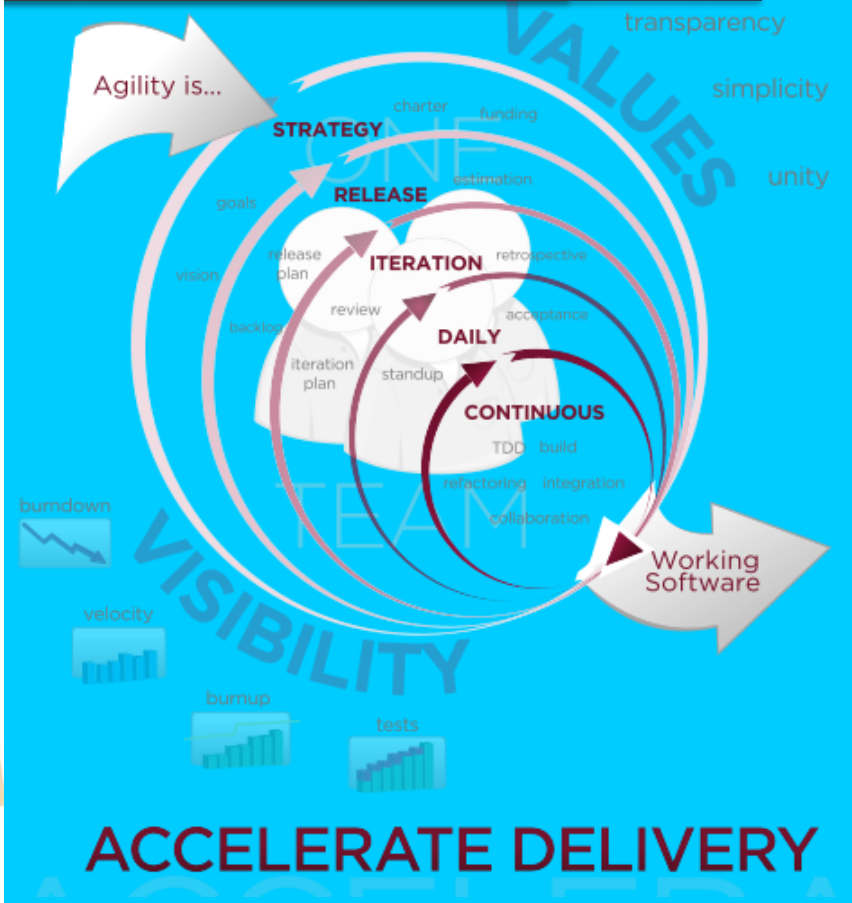
Use when there is reusable software you can integrate.

Types of reusable software

- Stand-alone application systems (sometimes called COTS) that are configured for use in a particular environment.
- Collections of objects that are developed as a package to be integrated with a component framework such as .NET or J2EE.
- Web services that are developed according to service standards and which are available for remote invocation.

The Agile Approach





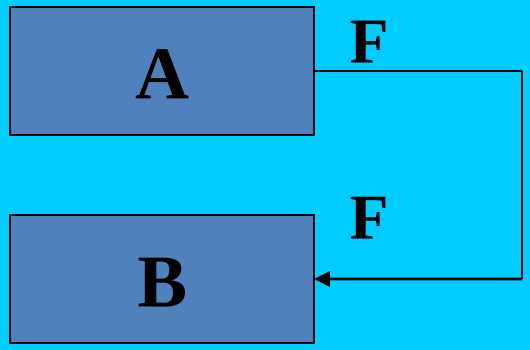
Agile is used ubiquitously

Project Scheduling

Scheduling Terminology continued…

Finish to Start (FS): A must finish before B can start.

Finish to Finish (FF): A must be finished before B can finish.

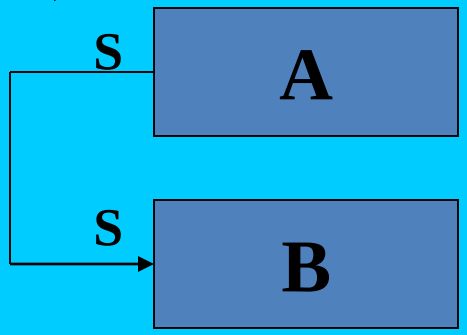


.

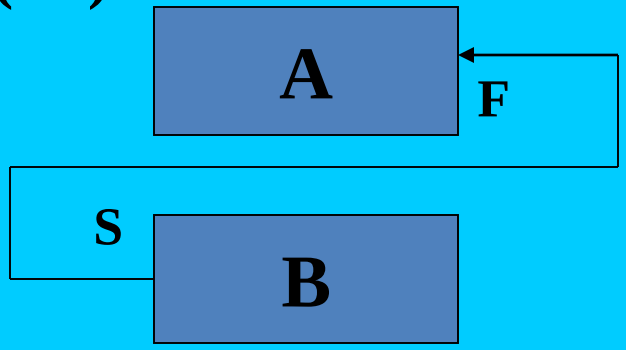
Scheduling Terminology continued…

Start to Start (SS): A must have started before B can start.

.

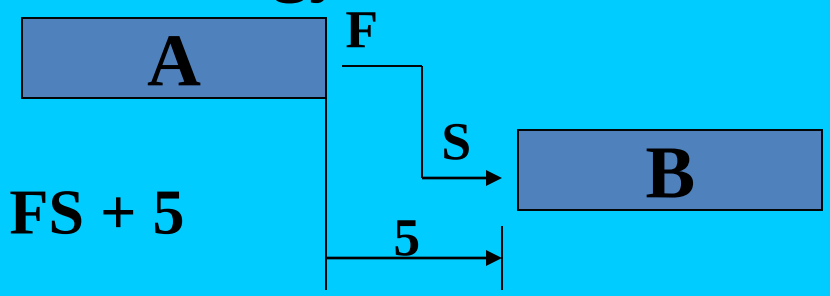


Start to Finish (SF): B must be started before A can finish.

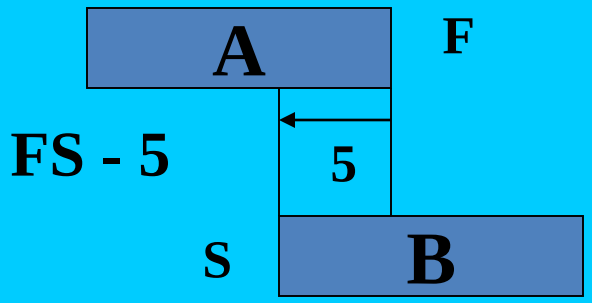


Scheduling Terminology continued…

Lag:



Lead:



Scheduling Terminology continued…

ASAP – As Soon As Possible
ALAP – As Late As Possible
SNET – Start No Earlier Than
FNET – Finish No Earlier Than
SNLT – Start No Later Than
FNLT – Finish No Later Than
MSO – Must Start On
MFO – Must Finish On

Scheduling Terminology continued…

Changing the Schedule:

Fast Track: scrutinize critical path for parallel activities. No cost implications.

Crashing: adding more resources to reduce duration – adding cost to reduce schedule.

Negotiation: Can you negotiate a change to schedule or scope? e.g. scope trade-offs – swapping one activity for another.

Project Scheduling

- ✧ Gantt Charts
  - Simple
  - Lends itself to end user communication
  - Drawn to scale
- ✧ PERT diagrams
  - Useful when activities can be done in parallel

PM Charts

Note: Break your project into hours, not weeks.

Example 1

Activity Chart

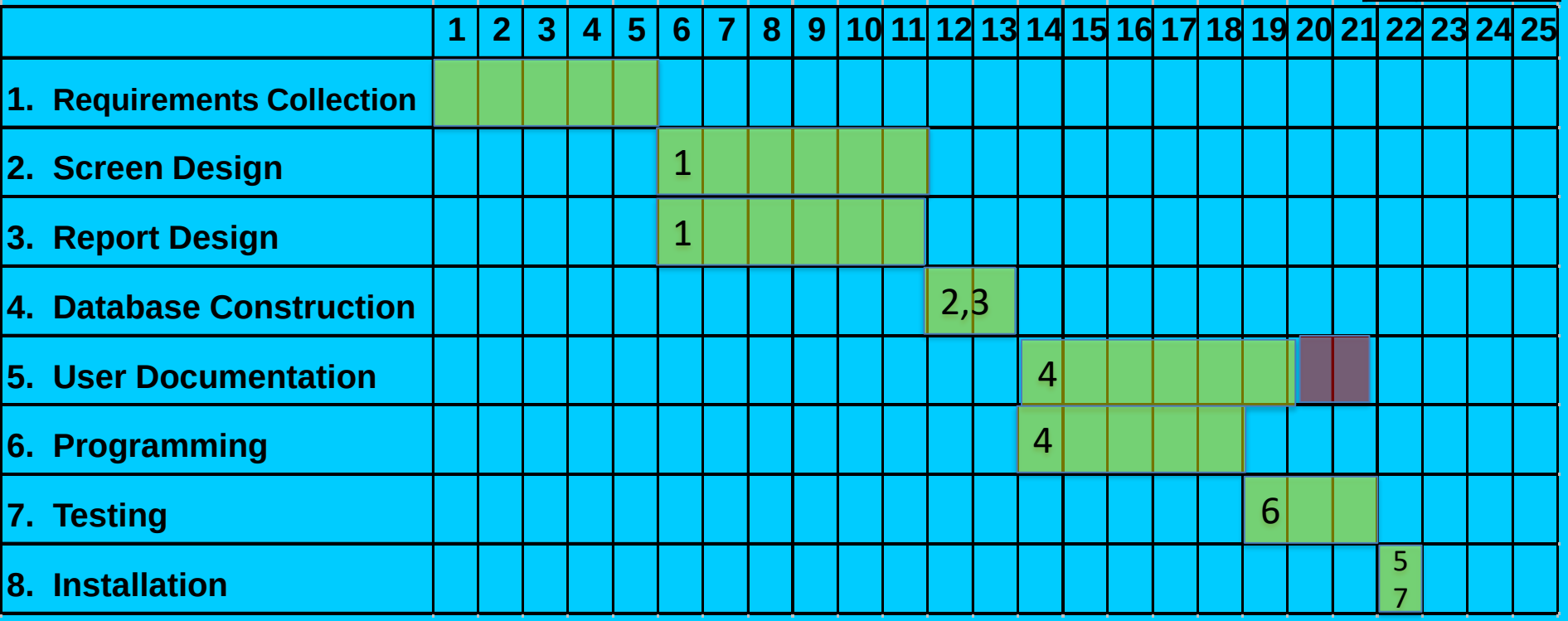
Task	Duration (PWks)	Predecessor Task(s)
1. Requirements Collection	5	-
2. Screen Design	6	1
3. Report Design	6	1
4. Database Construction	2	2, 3
5. User Documentation	6	4
6. Programming	5	4
7. Testing	3	6
8. Installation	1	5, 7

Project Management Charts

Example 1 continued…

1	5	-
2	6	1
3	6	1
4	2	2, 3
5	6	4
6	5	4
7	3	6
8	1	5, 7

Gantt Chart



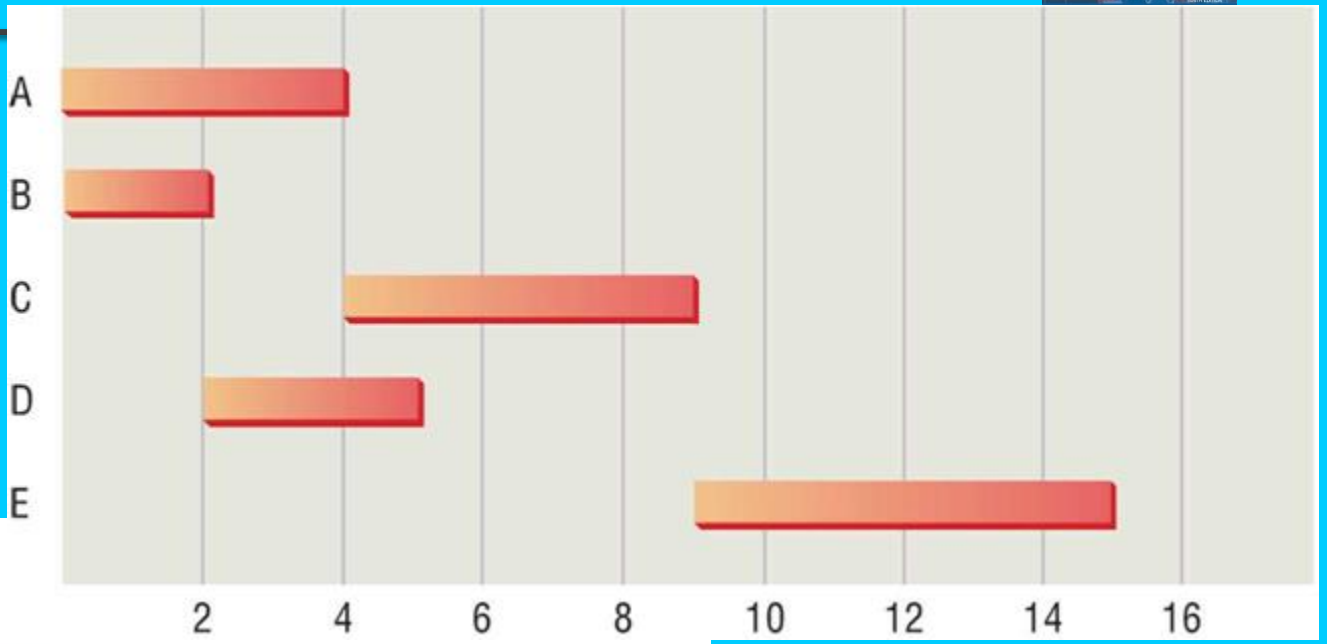
PM Charts

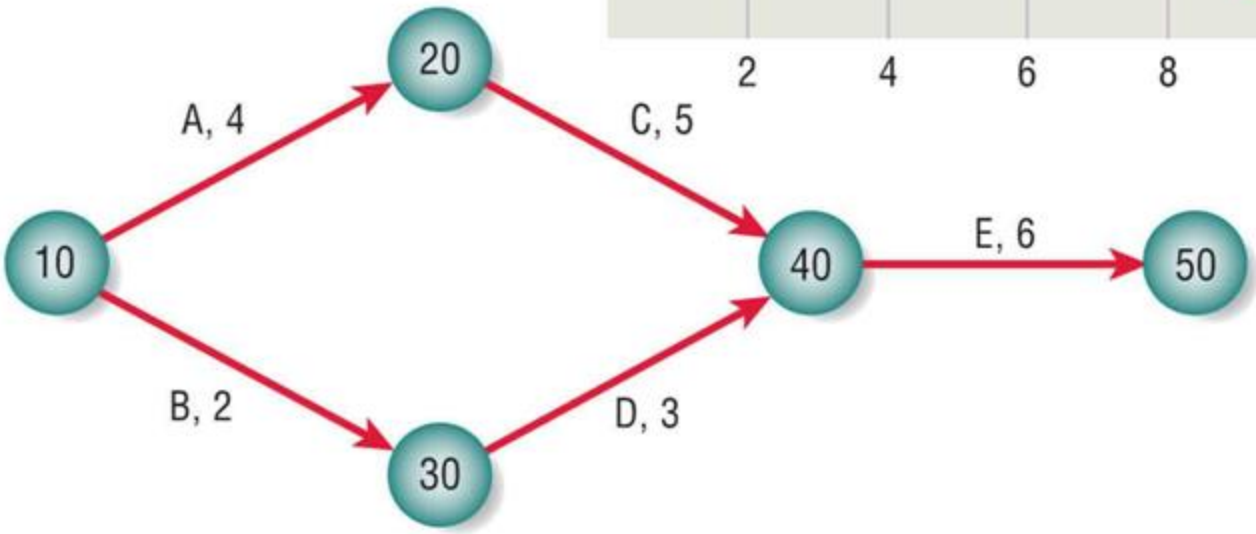
Example 1 continued…

Gantt Chart

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1. Requirements Collection																									
2. Screen Design						1																			
3. Report Design						1																			
4. Database Construction												2,3													
5. User Documentation														4											
6. Programming														4											
7. Testing																			6						
8. Installation																						5,7			

PERT Charts –(info on the edge)



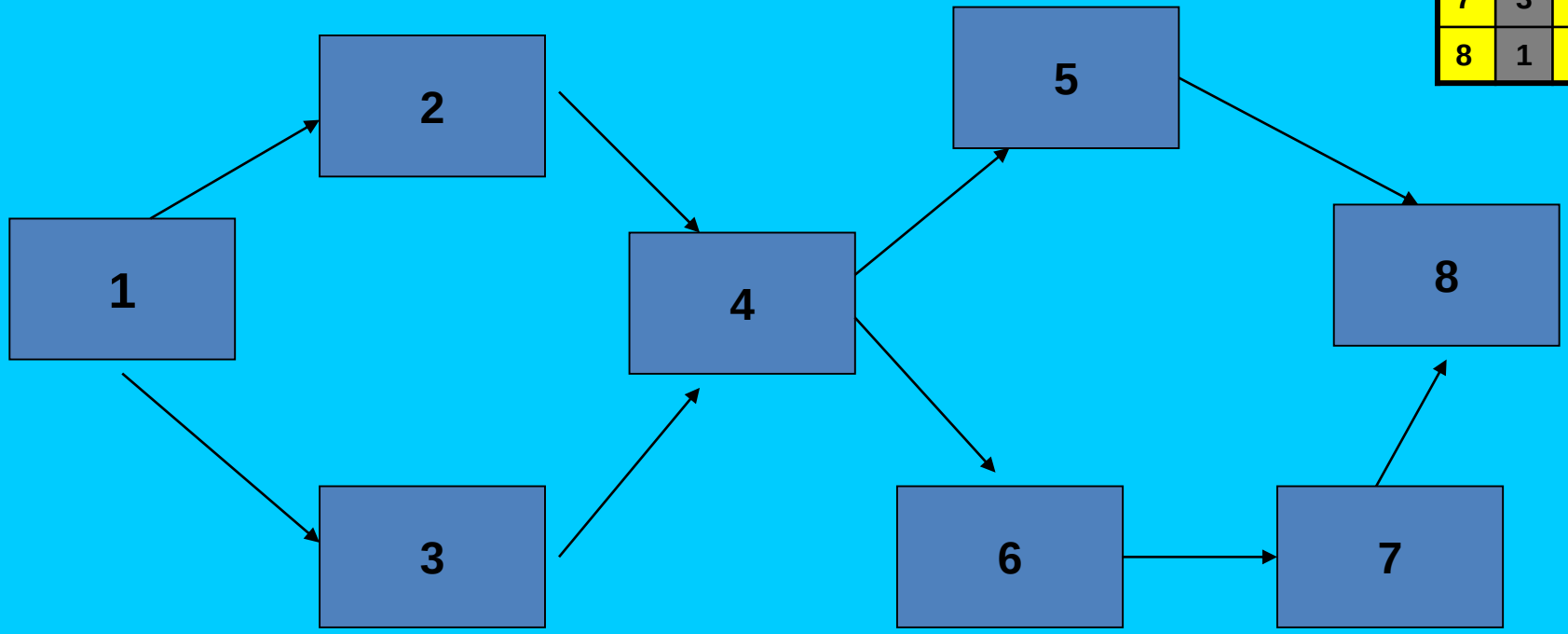


Copyright ©2011 Pearson Education, Inc. publishing as Prentice Hall

PM Charts: Network diag

Example 1 continued…

1	5	-
2	6	1
3	6	1
4	2	2, 3
5	6	4
6	5	4
7	3	6
8	1	5, 7



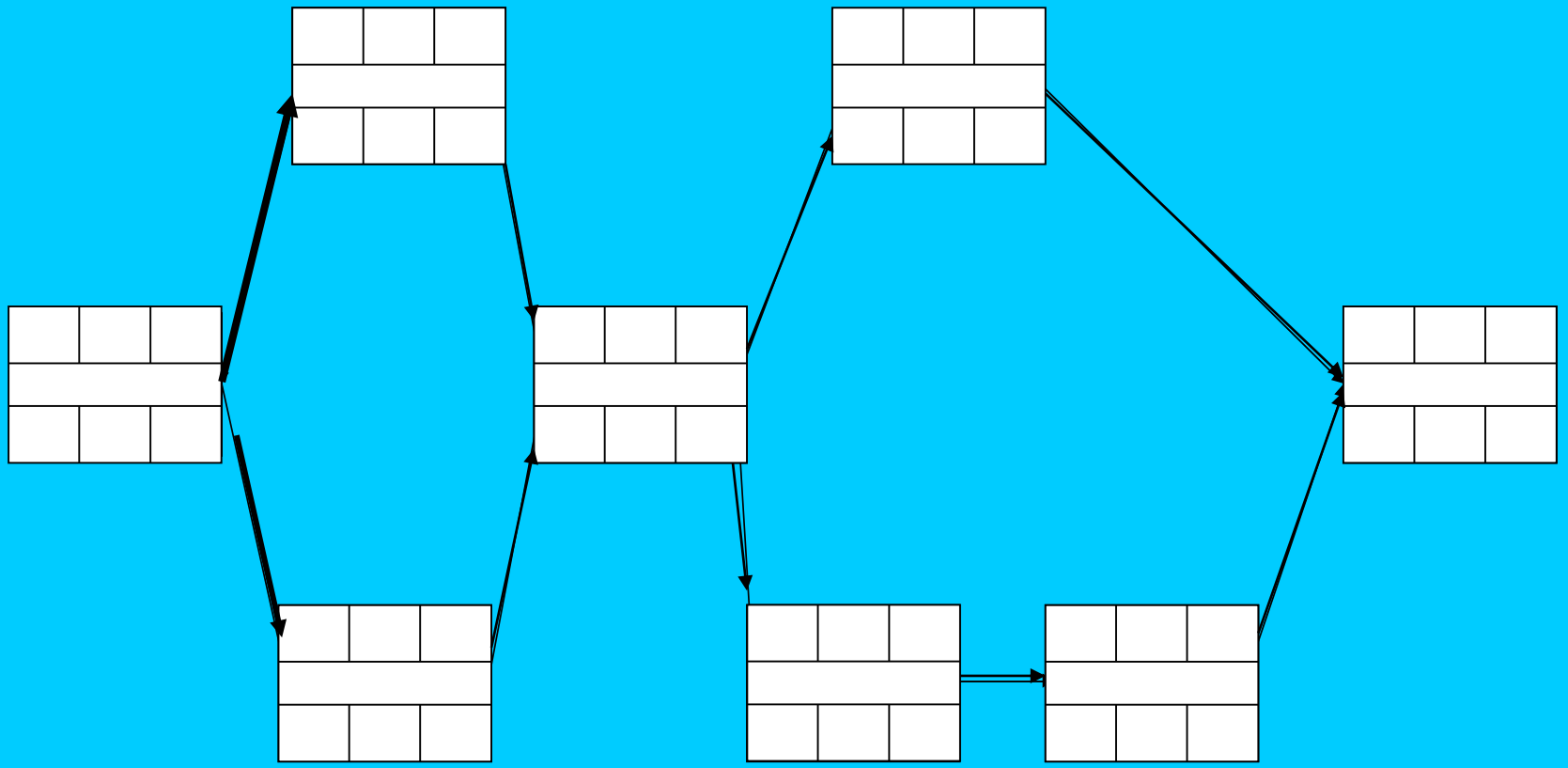
PM Charts: PERT

Example 1 continued…

Early Start ES	Duration	Early Finish EF
Task Name		
Late Start LS	Slack	Late Finish LF

PM Charts PERT

Example 1 continued…

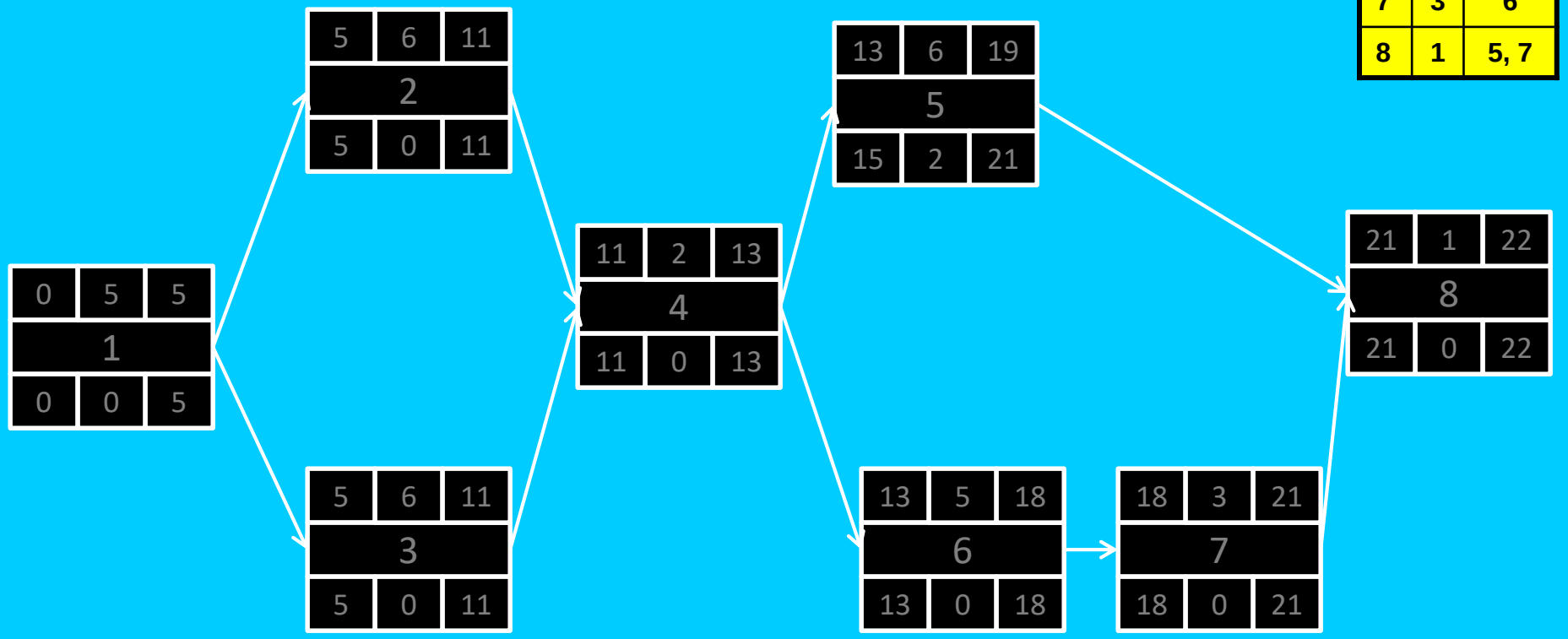


Project Management Charts

Example 1 continued…

ES	D	EF
Name		
LS	S	LF

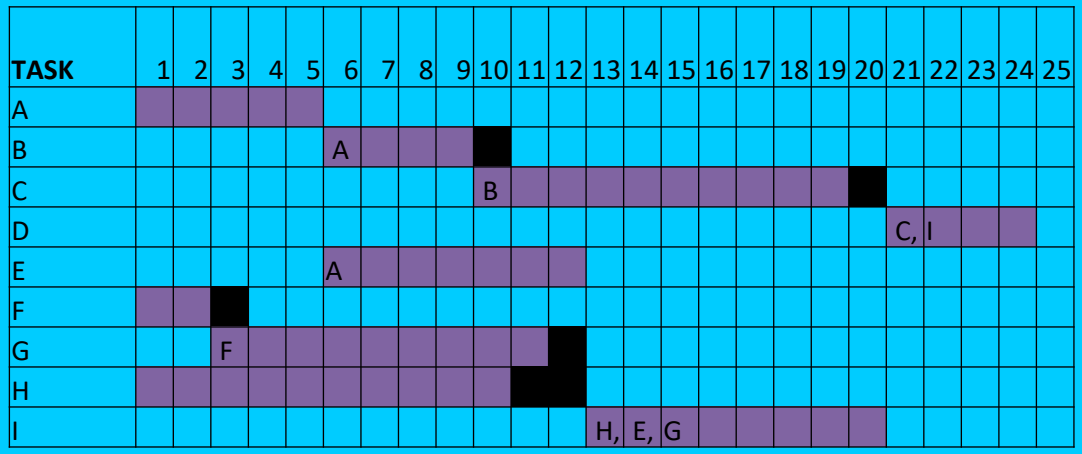
1	5	-
2	6	1
3	6	1
4	2	2, 3
5	6	4
6	5	4
7	3	6
8	1	5, 7



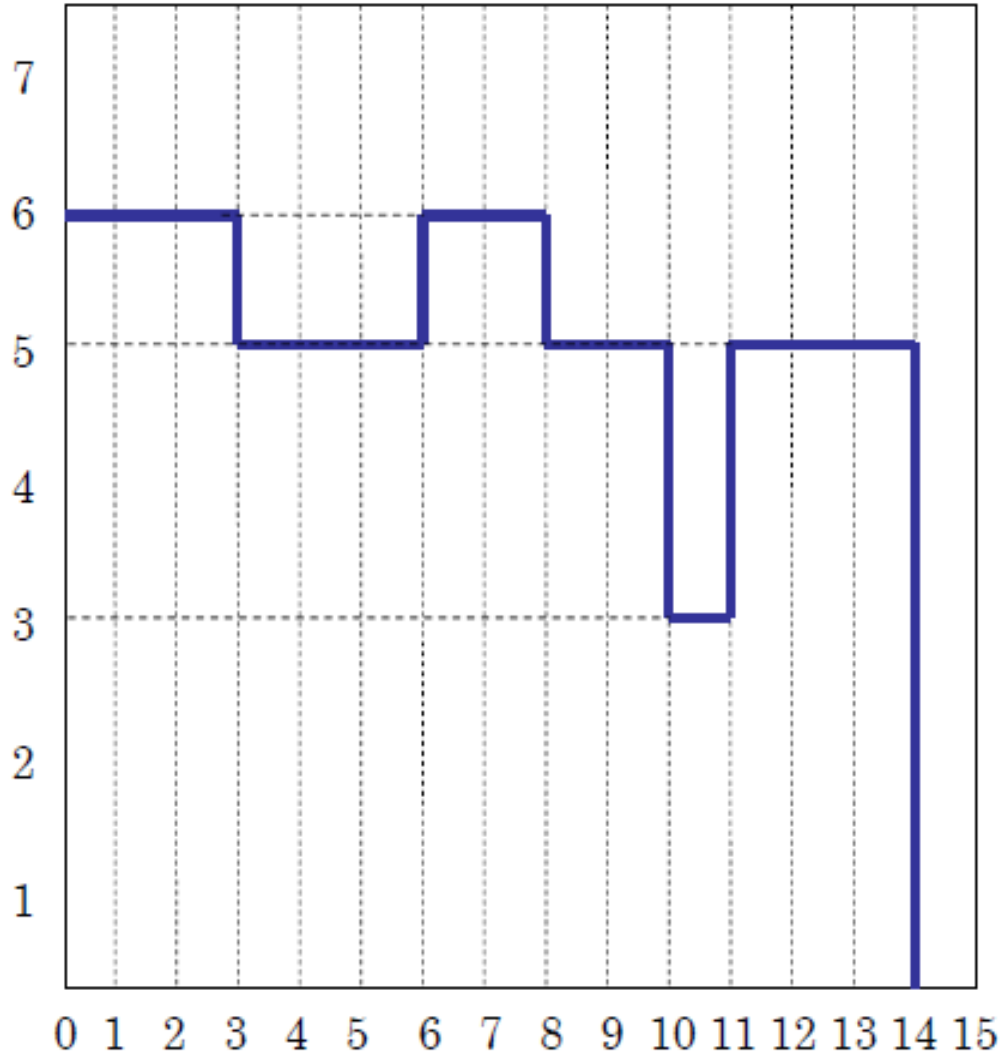
Try it yourself

TASK	Description	Duration (Days)	Must follow
A	Create use case diag	5	None
B	Create use case scenario	4	A
C	Revise diag and scenario	10	B
D	Write use case	4	C, I
E	Undertake Cost Benefit	7	A
F	Do report interface	2	None
G	Review report interface	9	F
H	Write charter	10	None
I	Create ER diagram	8	H, E, and G

What does the Pert/Gantt chart look like?



Pert and Gant Together



Task	Duration	Required Staff
A	3	2
B	4	4
C	5	1
D	7	3
E	2	1
F	4	2
G	3	5