SS ZG622: Software Project Management (Lecture #7)

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

T V Rao, BITS-Pilani Off-campus Centre, Hyderabad

Text Books

T1: Bob Hughes, Mike Cotterel, and Rajib Mall, Software Project Management, 5th Edition, McGraw Hill, 2011

T2: Pressman, R.S. Software Engineering : A Practitioner's Approach, 7th Edition, McGraw Hill, 2010

R1: Sommerville, I., Software Engineering, Pearson Education, 9th Ed., 2010

R2: Capers Jones., Software Engineering Best Practices, TMH ©2010

R3: Robert K. Wysocki, Effective Software Project Management, John Wiley & Sons © 2006

R4: George Stepanek, Software Project Secrets : Why Software Projects Fail, Apress ©2012

R5: A Guide to the Project Management Body of Knowledge (PMBOK® Guide), Fifth Edition by Project Management Institute Project Management Institute © 2013

R6: Jake Kouns and Daniel Minoli, Information Technology Risk Management in Enterprise Environments. John Wiley & Sons © 2010

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

L7: Sotware Project Planning –

Activity and Resource Planning, Risk Management

Source Courtesy: Some of the contents of this PPT are sourced from materials provided by publishers of prescribed books

Software Project Estimation (Review)

- Planning requires technical managers and the software team to make an estimation that determines how much money, effort, resources, and time it will take to build a specific system or product
- Process and project metrics can provide a historical perspective and valuable input for generation of quantitative estimates
- Past experience can aid greatly
- Estimation carries inherent risk, and this risk leads to uncertainty. Risk increases with
  - Project Complexity
  - Project Size
  - The degree of structural uncertainty
- The availability of dependable historical information has a strong influence on estimation risk
- A project manager should not become obsessive about estimation
  - Plans should be iterative and allow adjustments as time passes and more is made certain

"It is the mark of an instructed mind to rest satisfied with the degree of precision that the nature of the subject admits, and not to seek exactness when only an approximation of the truth is possible." ARISTOTLE

Feasibility

- After the project scope is adequately understood, feasibility is addressed
- Software feasibility has four dimensions, (as per Putnam & Meyer)
  - **Technology** – Is the project technically feasible? Is it within the state of the art? Can defects be reduced to a level matching the application's needs?
  - **Finance** – Is is financially feasible? Can development be completed at a cost that the organization, its client, or the market can afford?
    - *Software Services Organization perspective*: Can we do the project profitably?
  - **Time** – Will the project's time-to-market beat the competition?
    - *Software Services Organization perspective*: Can we meet timelines as agreed in contract?
  - **Resources** – Does the software organization have the resources needed to succeed in doing the project?

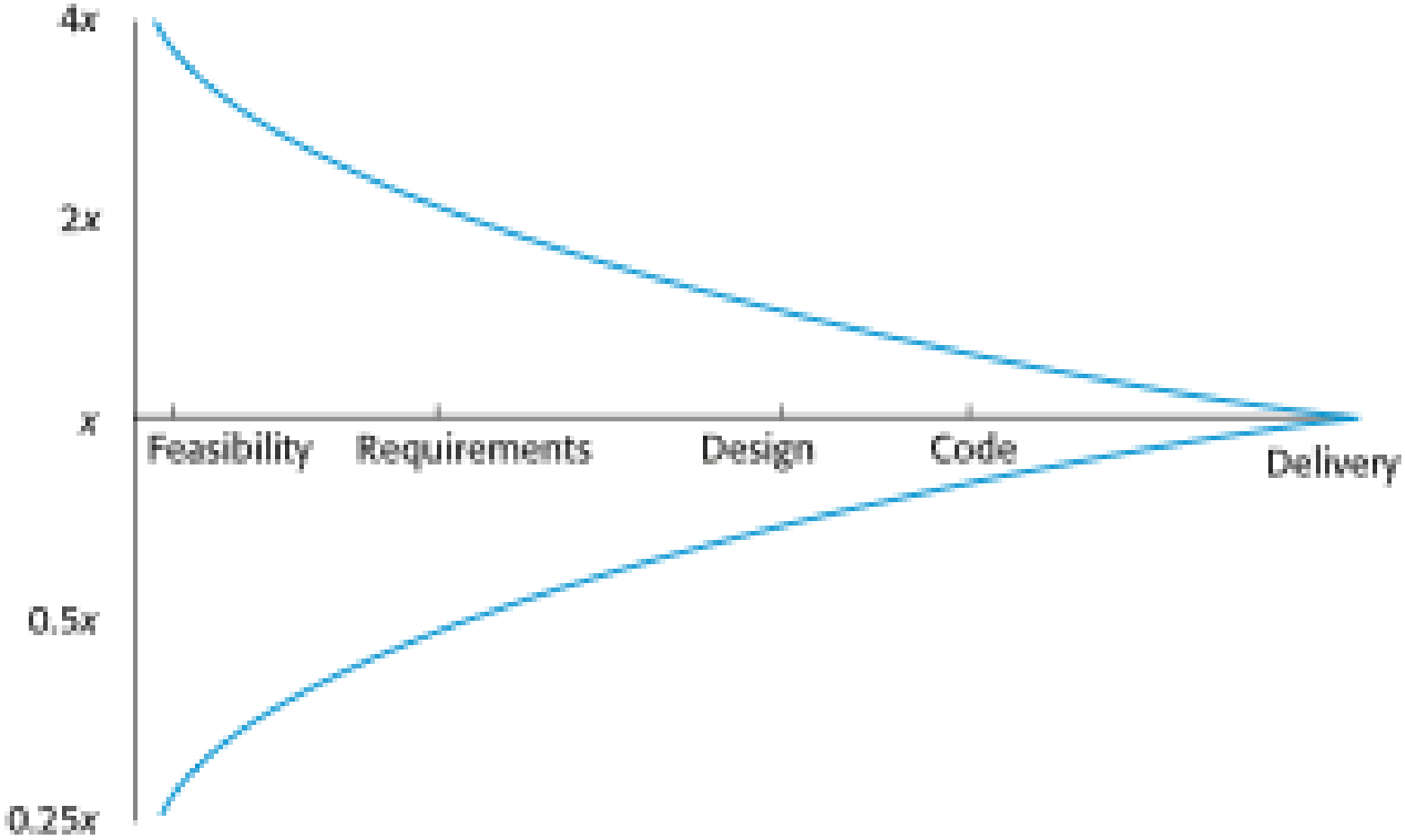
Another view recommends the following feasibility dimensions: technological, economical, **legal**, **operational**, and schedule issues (TELOS)

Estimation for Web Applications

Roetzheim recommends a modified FP approach:

- *Inputs* are each input screen or form, each maintenance screen, and if you use a tab notebook metaphor each tab.
- *Outputs* are each static Web page, each dynamic web page script and each report
- *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).
- *Interfaces* retain their definition as logical files into our out-of-the-system boundaries.
- *Queries* are each externally published or use a message-oriented interface.

Estimate uncertainty



Sommerville, I., Software Engineering, Pearson Education, 9th Ed., 2010

Estimate uncertainty

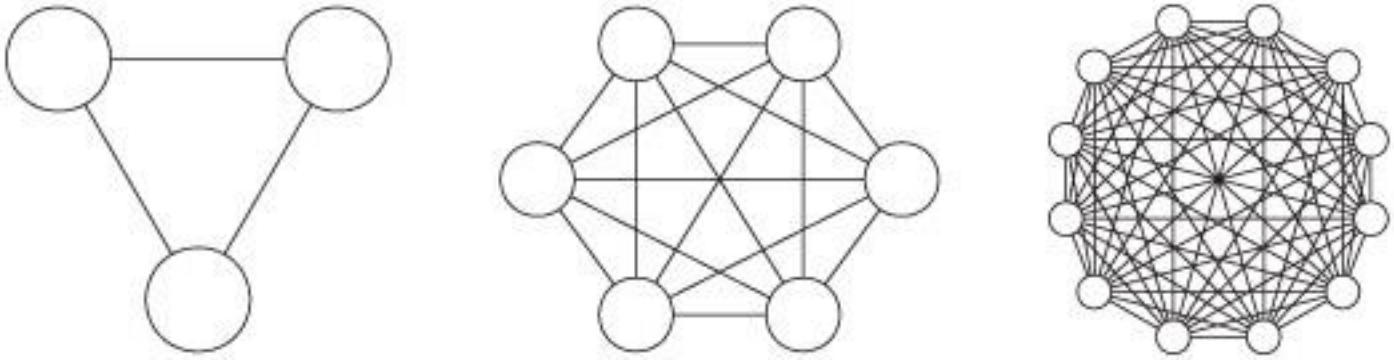
Accuracies of Estimates in Other Industries vs. Software Development

Project Management Accuracy from the PMBOK Third Edition [PMI 2004]		Software Development Accuracy from Rapid Development [McConnell 1996]	
Conceptual	30% under to 50% over	75% under to 300% over	Initial product concept
Preliminary	20% under to 30% over	50% under to 100% over	Approved product definition
Definitive	15% under to 20% over	33% under to 50% over	Requirements specification
Control	10% under to 15% over	20% under to 25% over	Product design specification

Software Challenges – Team Size

Large programming projects suffer management problems that are qualitatively different than small ones because of the division of labour; that the conceptual integrity of the software product is thus critical; and that it is difficult but possible to achieve this conceptual integrity.

- *The Mythical Man-Month (Fred Brooks)*



No of communication paths explode as team size grows. Above illustrations show communication paths among teams of 3, 6, and 12

Software Challenges – Productivity

There are order-of-magnitude differences among programmers and have been confirmed by many … studies of professional programmers….

- Code Complete (Steve McConnell)

Barry Boehm [1981] makes the following recommendations for selecting team in his book *Software Engineering Economics*:

Top talent—Use better and fewer people.

Job matching—Fit the tasks to the skills and motivation of the people available.

Team balance—Select people who will complement and harmonize with each other.

Basic Principles for Project Scheduling

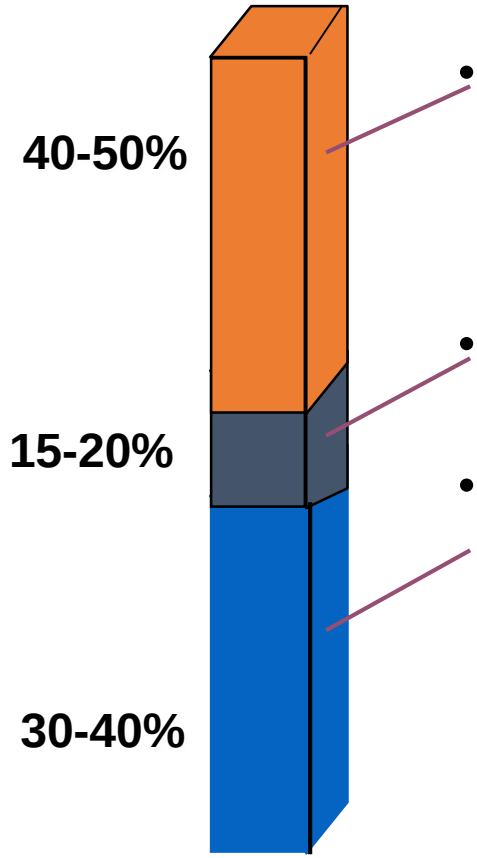
- Compartmentalization
  - The project must be compartmentalized into a number of manageable activities, actions, and tasks; both the product and the process are decomposed
- Interdependency
  - The interdependency of each compartmentalized activity, action, or task must be determined
  - Some tasks must occur in sequence while others can occur in parallel
  - Some actions or activities cannot commence until the work product produced by another is available
- Time allocation
  - Each task to be scheduled must be allocated some number of work units
  - In addition, each task must be assigned a start date and a completion date that are a function of the interdependencies
  - Start and stop dates are also established based on whether work will be conducted on a full-time or part-time basis

(More on next slide)

Basic Principles for Project Scheduling (continued)

- Effort validation
  - Every project has a defined number of people on the team
  - As time allocation occurs, the project manager must ensure that no more than the allocated number of people have been scheduled at any given time
- Defined responsibilities
  - Every task that is scheduled should be assigned to a specific team member
- Defined outcomes
  - Every task that is scheduled should have a defined outcome for software projects such as a work product or part of a work product
  - Work products are often combined in deliverables
- Defined milestones
  - Every task or group of tasks should be associated with a project milestone
  - A milestone is accomplished when one or more work products has been reviewed for quality and has been approved

Software 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

A word on Film Production

40 20 40 'Guide' for project time and resources:

40% **Pre-Production** Planning

20% **Production** Filming

40% **Post-production** - editing, reflecting, promoting, screening

-Ian McCormick, Former Professor in Arts, University of Northampton

Activity planning

Activity Networks

- A project is:
  - Composed of a number of **activities**
  - May start when at least one of its activities is ready to start
  - Completed when all its activities are completed

- Activity Networks help us to:
  - Assess the feasibility of the planned project completion date
  - Identify when resources will need to be deployed to activities
  - Calculate when costs will be incurred

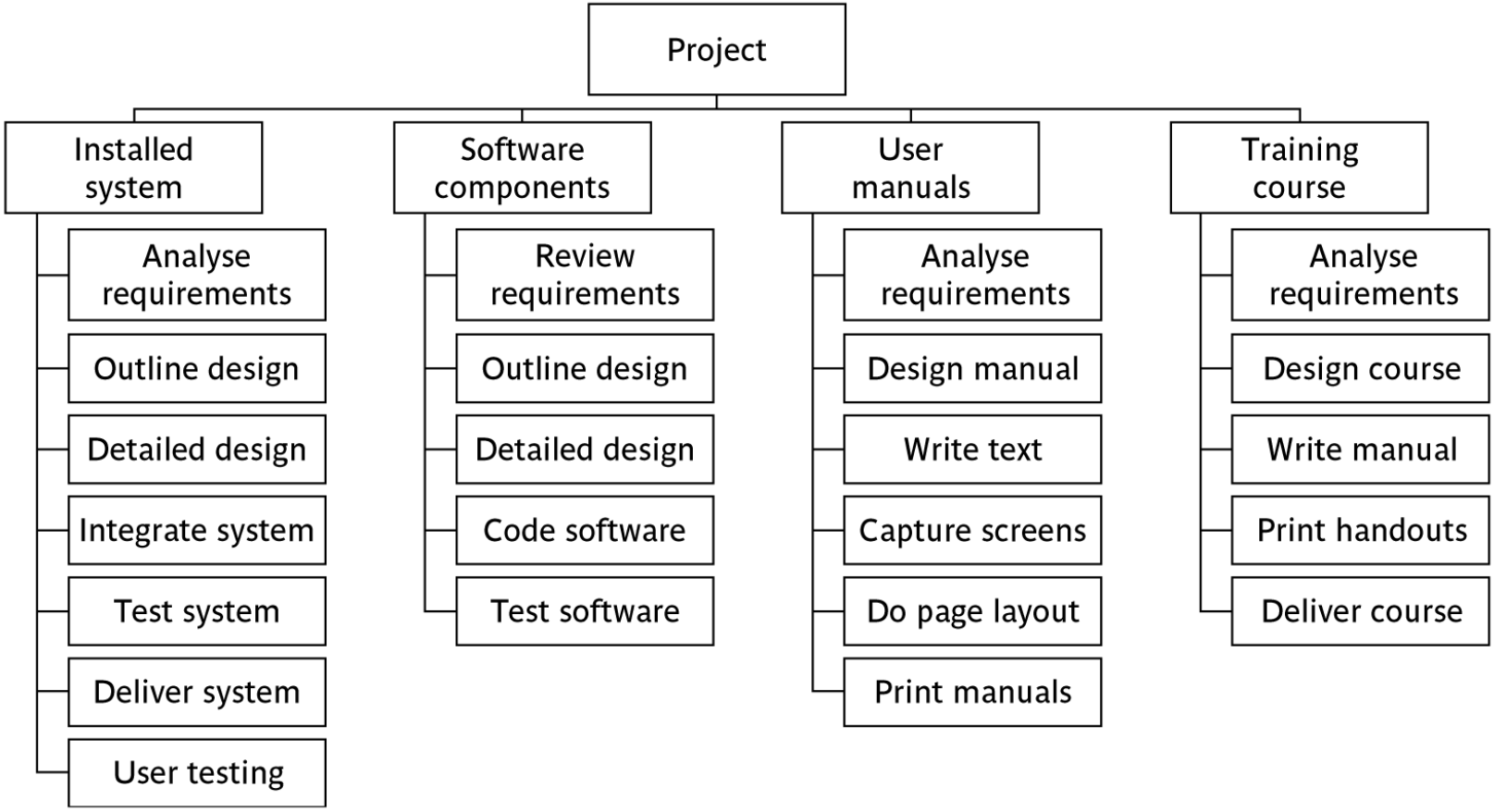
Defining activities

- An activity
  - Must have clearly defined start and end-points
  - Must have resource requirements that can be forecast: these are assumed to be constant throughout the project
  - Must have a duration that can be forecast
  - May be dependent on other activities being completed first (precedence networks)

Identifying activities

- Process-based approach
  - Work-based: draw-up a Work Breakdown Structure listing the work items needed
- Product-based approach
  - list the deliverable and intermediate products of project – product breakdown structure (PBS)
  - Identify the order in which products have to be created
  - work out the activities needed to create the products

Hybrid approach

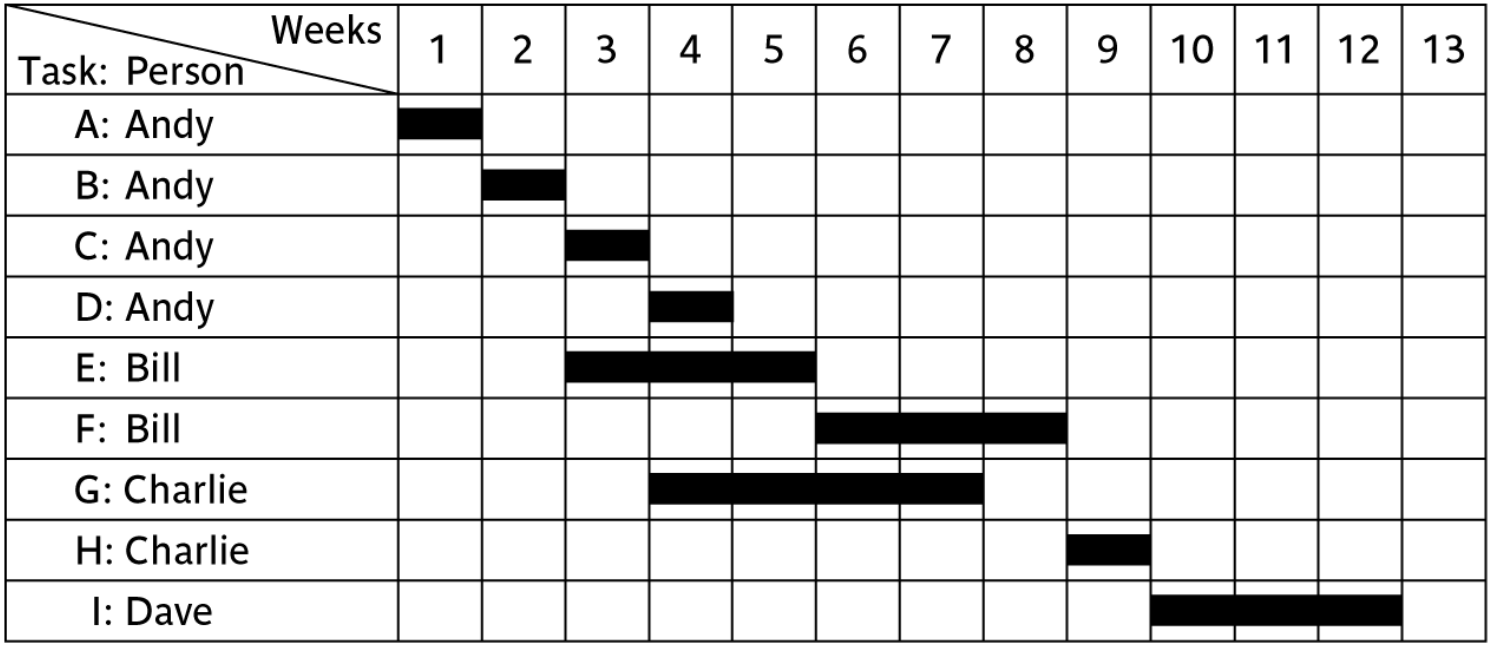


Objectives of Activity Planning

- Feasibility Assessment
- Resource Allocation
- Detailed Costing
- Motivation (Set targets for the team)
- Coordination

The final outcome of the planning process

A project plan as a bar chart

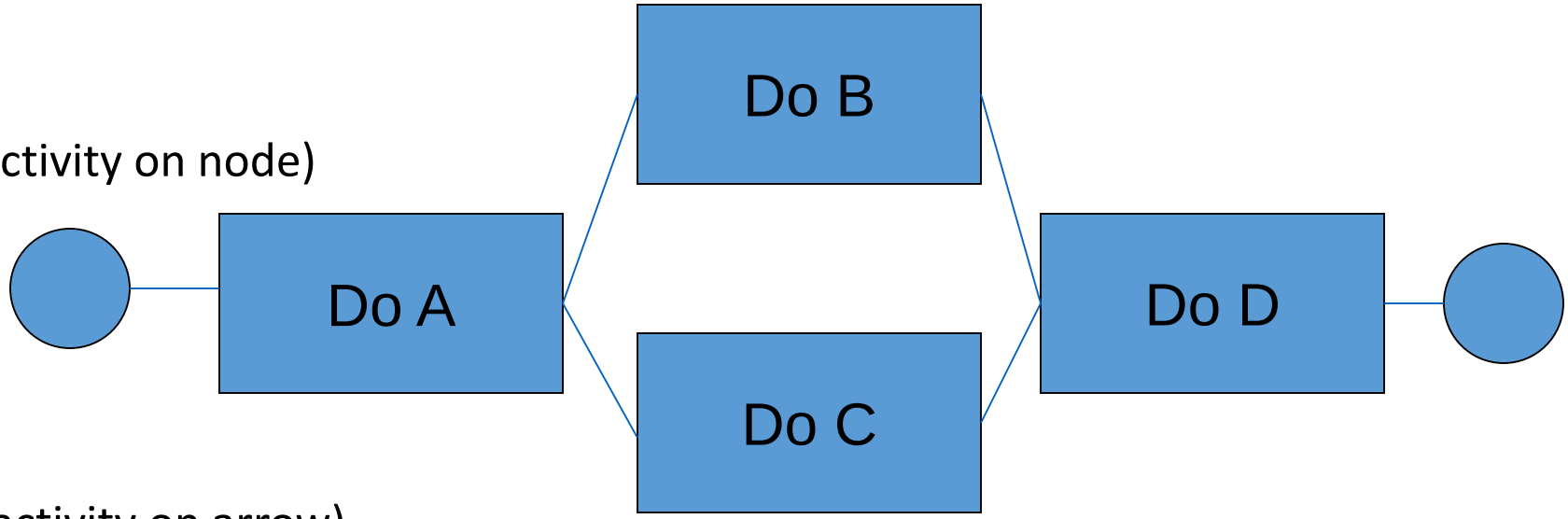


Activity key

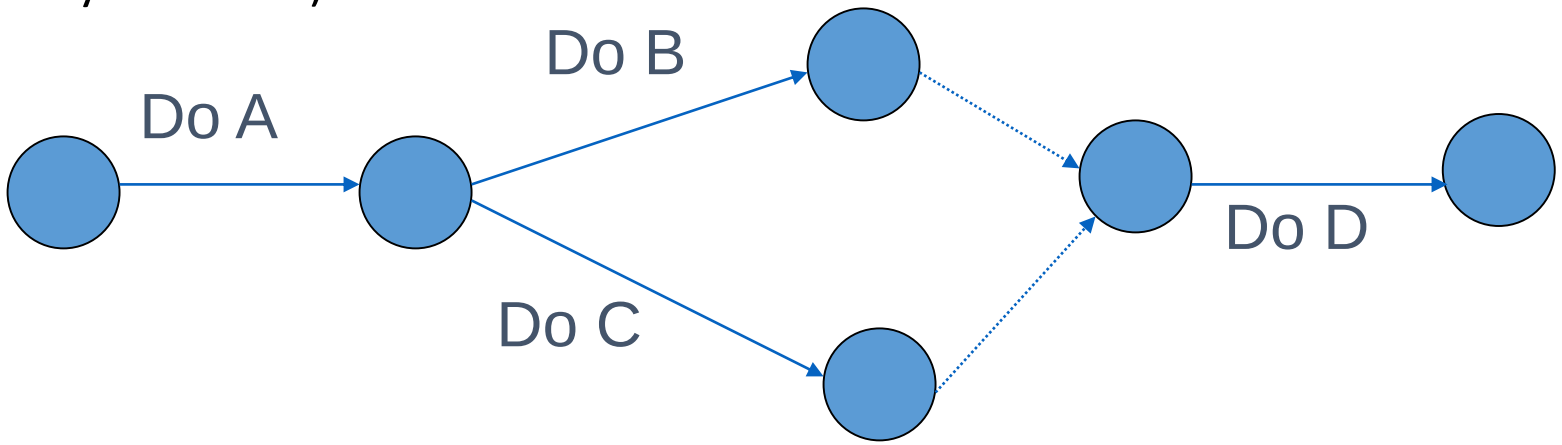
A: Overall design
B: Specify module 1
C: Specify module 2
D: Specify module 3
E: Code module 1
F : Code module 3
G: Code module 2
H: Integration testing
I : System tesing

Activity Network Diagrams (aka PERT/CPM charts)

AON (activity on node)

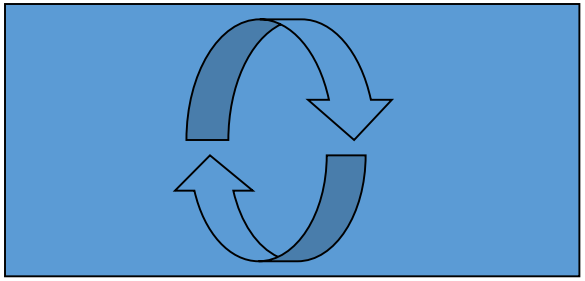


AOA(activity on arrow)



Drawing up a PERT Chart

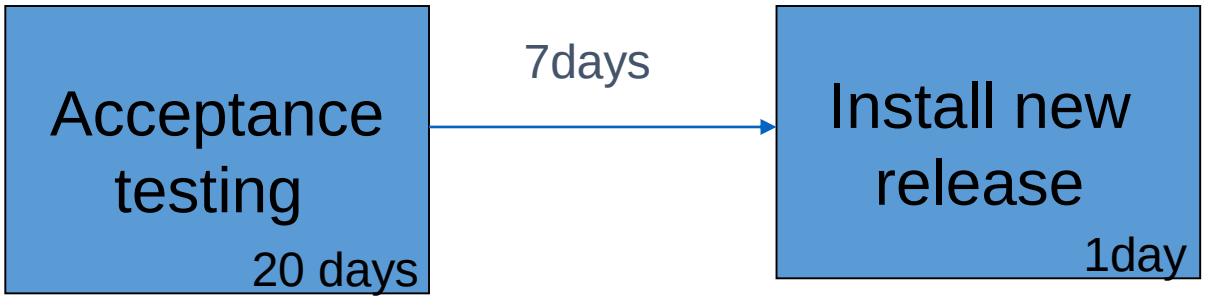
- **No** looping back is allowed – deal with iterations by hiding them within single activities



- *milestones* – 'activities', such as the start and end of the project, which indicate transition points. They have zero duration.

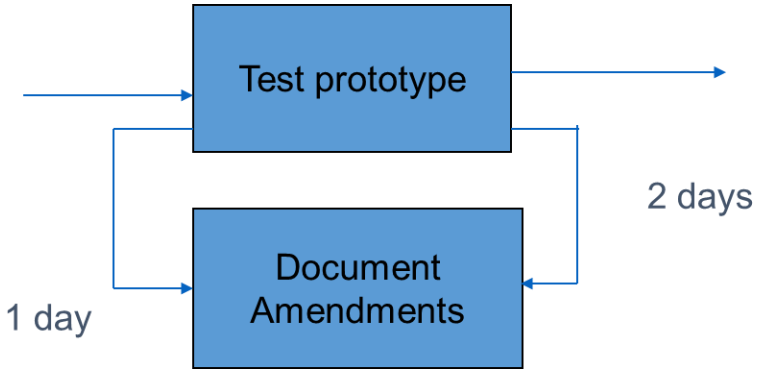
Lagged activities

- where there is a fixed delay between activities e.g. seven days notice has to be given to users that a new release has been signed off and is to be installed

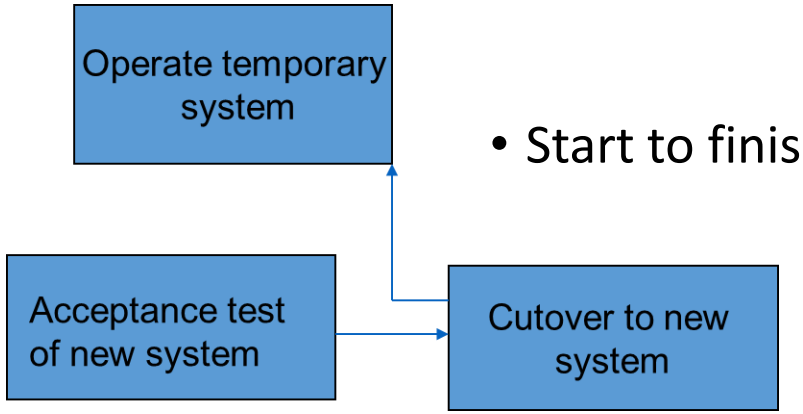


Types of links between activities

- Finish to start



- Start to start/ Finish to finish



- Start to finish

Start and finish times

- e.g. Activity ‘develop report software’
- Earliest start (ES)
- Earliest finish (EF) = ES + duration
- Latest finish (LF) = latest task can be completed without affecting project end Latest start = LF - duration

Example

- earliest start = day 5
- latest finish = day 30
- duration = 10 days

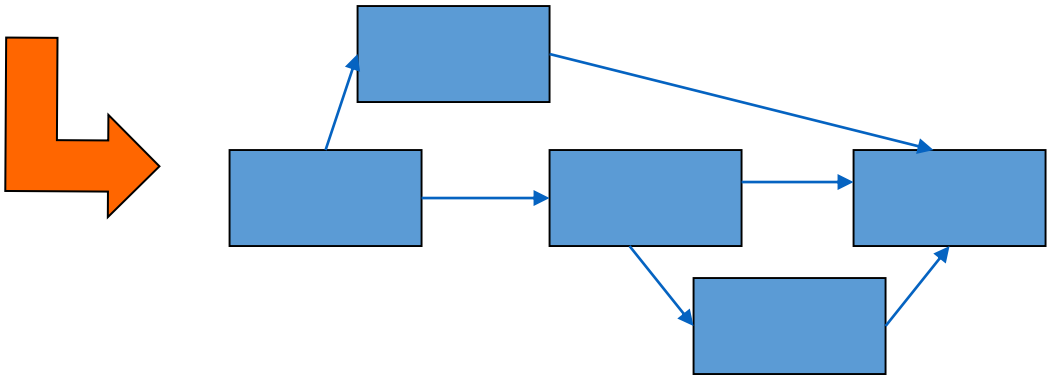
- earliest finish =
- latest start =

Float = LF - ES – duration = 15

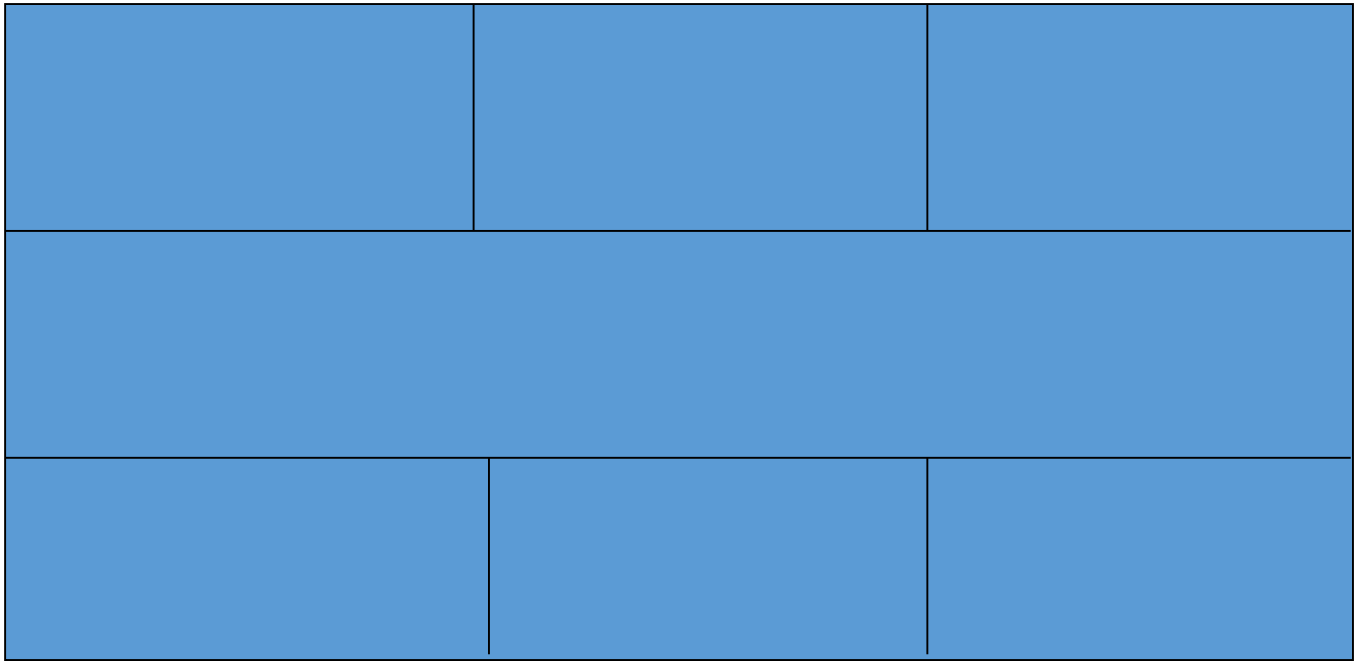
- Note that day numbers used rather than actual dates
- Makes initial calculations easier – not concerned with week-ends and public holidays
- For **finish** date/times Day 1 means at the END of Day 1.
- For a **start** date/time Day 1 also means at the END of Day 1.
- The first activity therefore begin at Day 0 i.e. the end of Day 0 i.e. the start of Day 1

Notation

Earliest start	Duration	Earliest finish
Activity label, activity description		
Latest start	Float	Latest finish

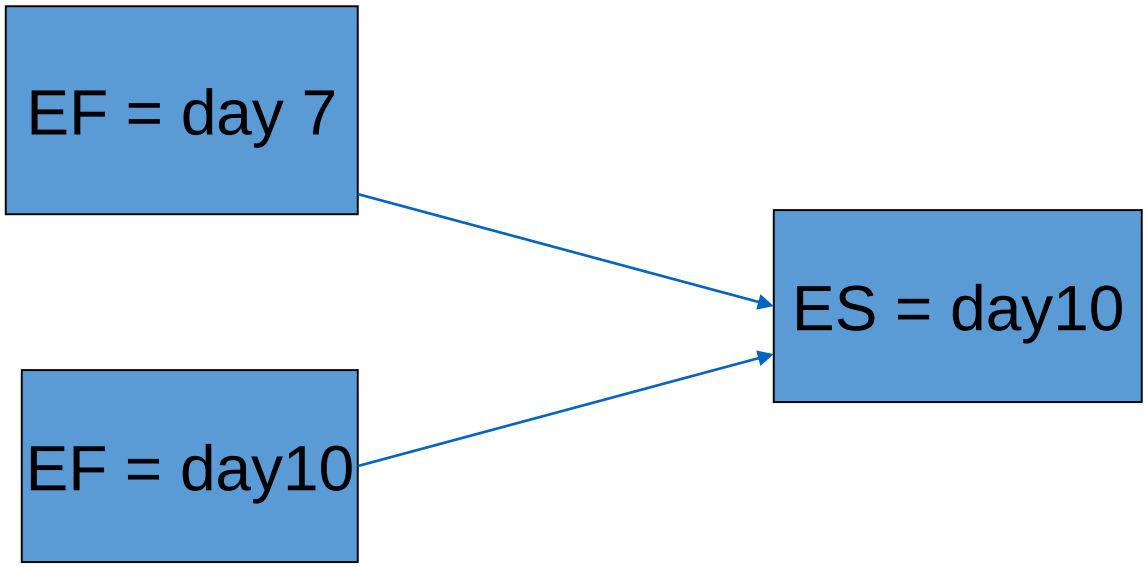


For the previous example

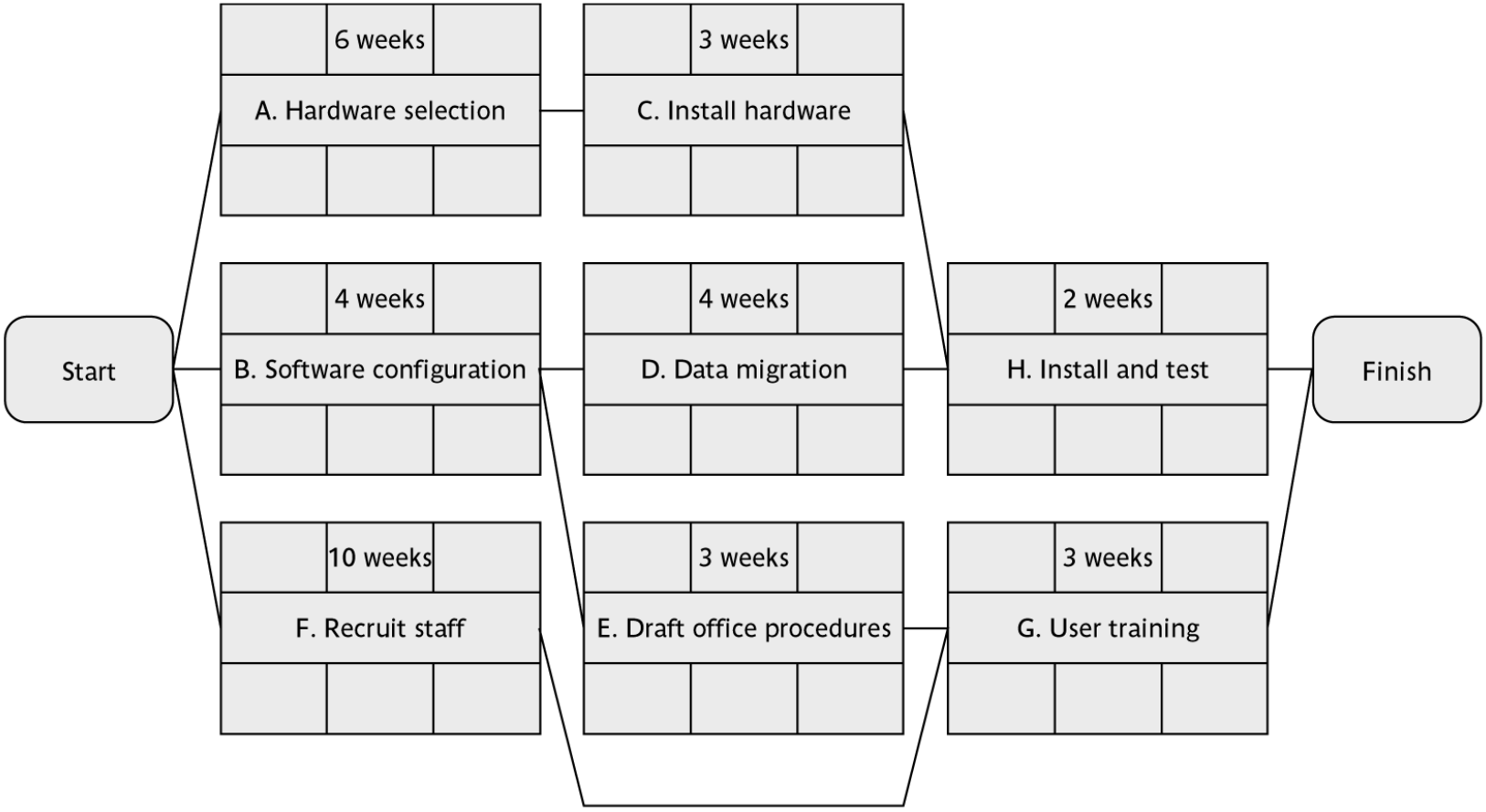


Forward pass

- Start at beginning (Day 0) and work forward following chains.
- Earliest start date for the *current* activity = earliest finish date for the *previous*
- When there is more than one previous activity, take the *latest* earliest finish



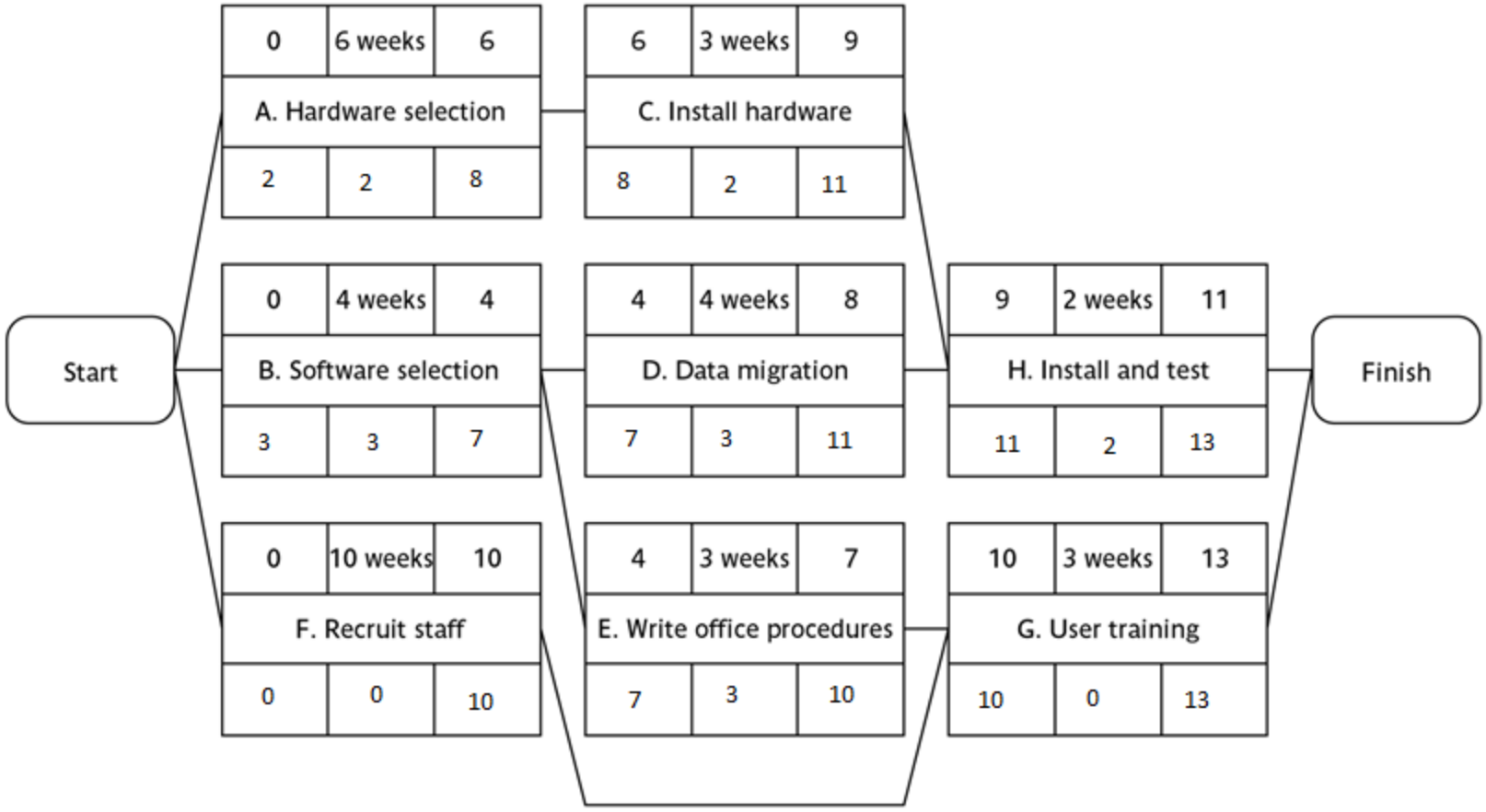
Example of an activity network



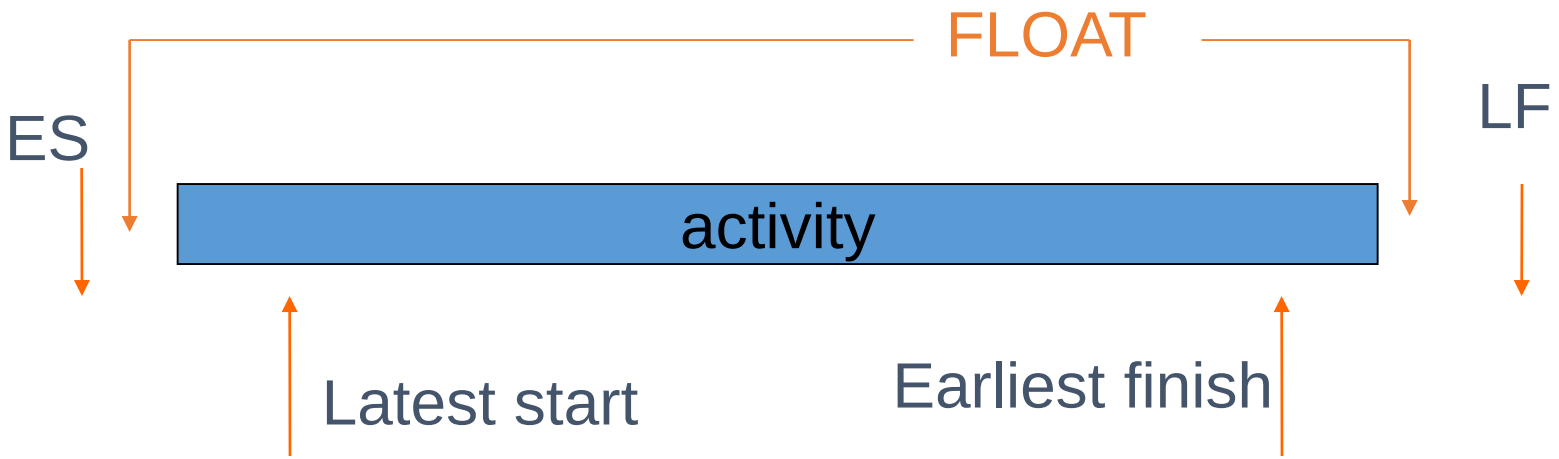
Backward pass

- Start from the *last* activity
- Latest finish (LF) for last activity = earliest finish (EF)
- work backwards
- Latest finish for *current* activity = Latest start for the *following*
- More than one following activity - take the *earliest* LS
- Latest start (LS) = LF for activity - duration

Completed Chart



Float

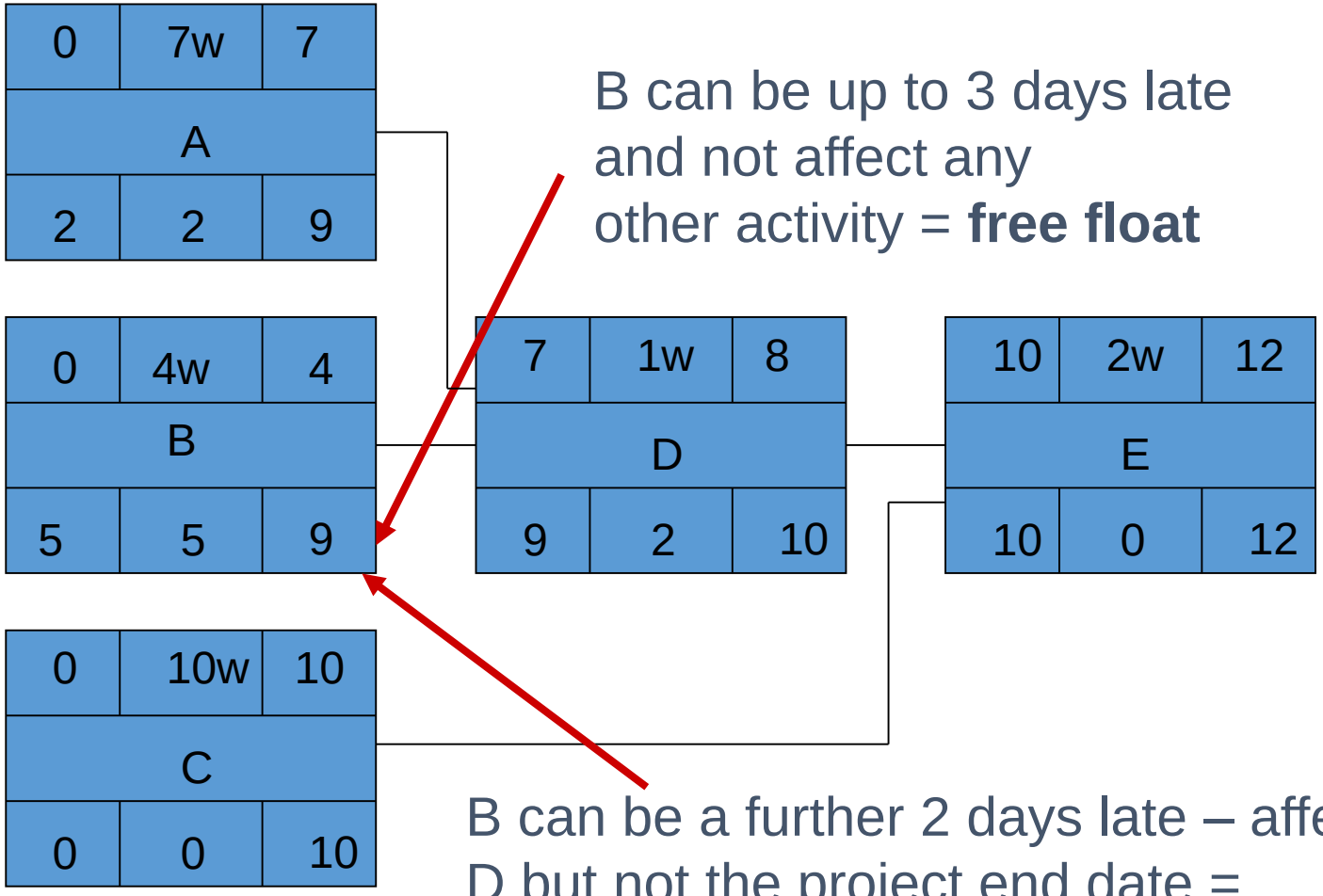


Float = Latest finish - Earliest start - Duration

Critical path

- Note the path through network with zero floats
- Critical path: any delay in an activity on this path will delay whole project
- Can there be more than one critical path?
- Sub-critical paths

Free and interfering float

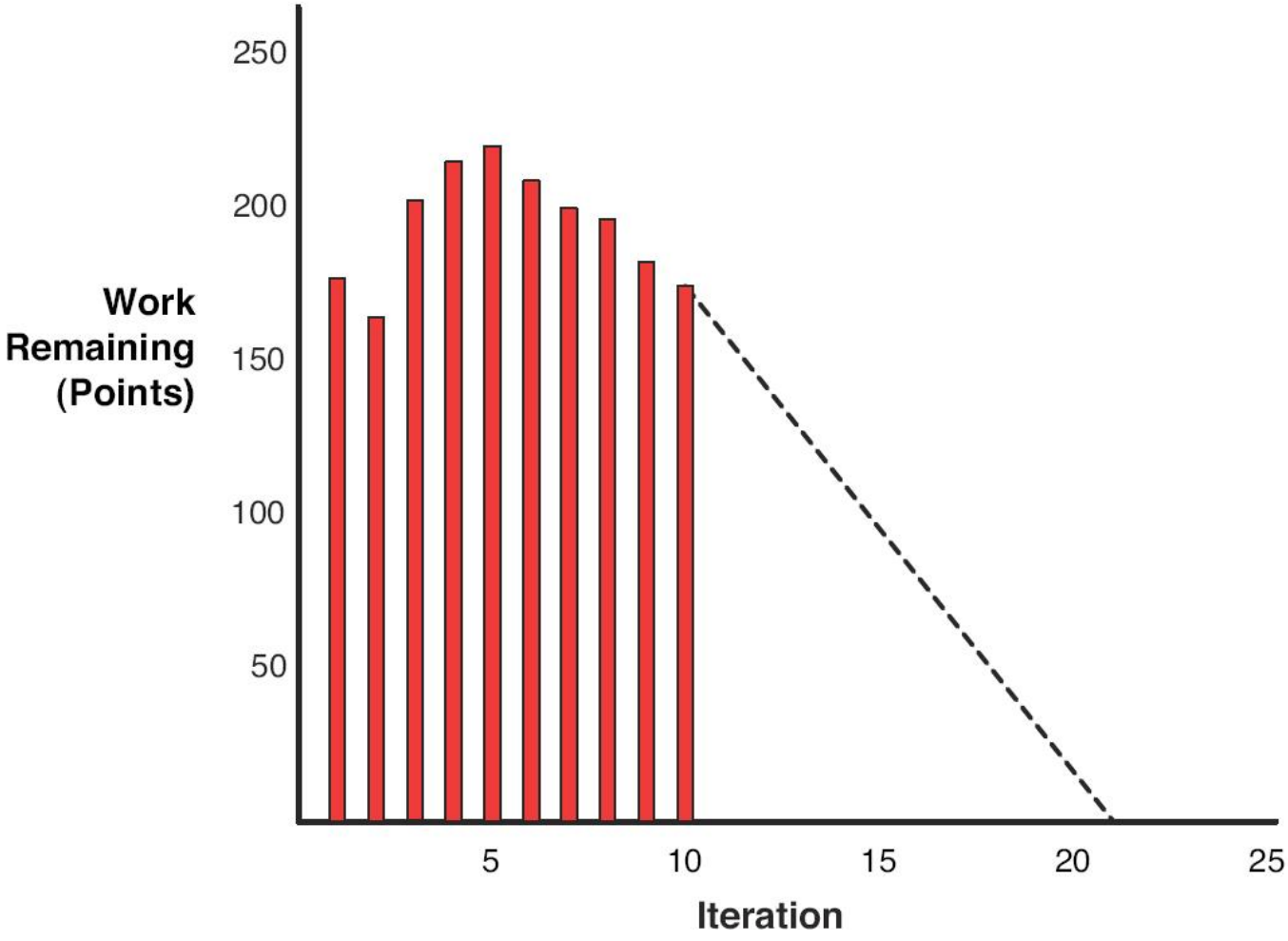


Agile Schedules

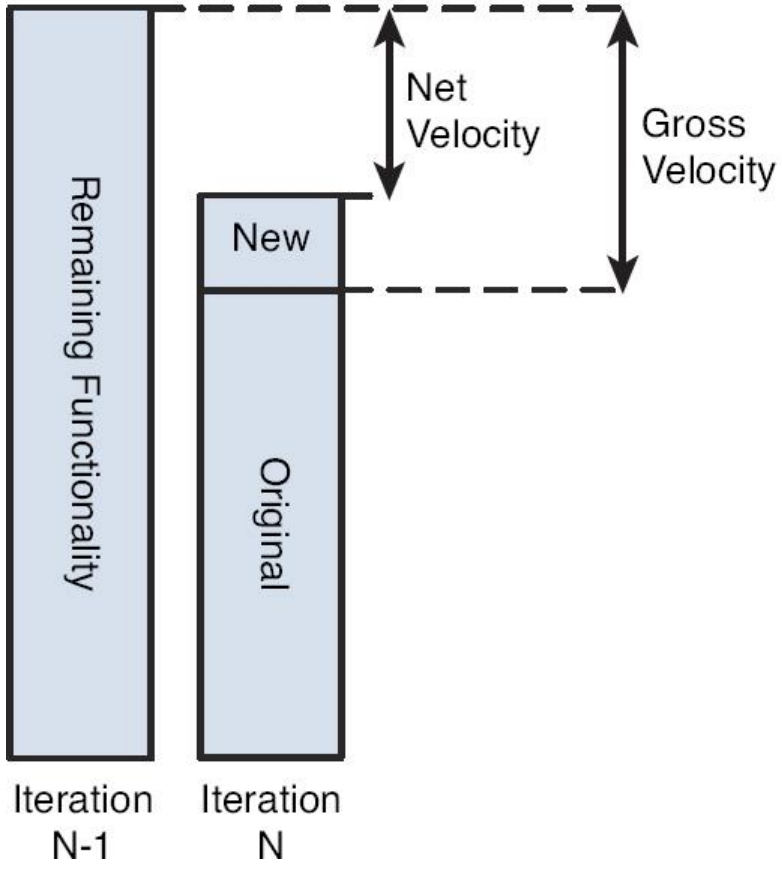
Time Boxing

- Severe deadline pressure may require the use of time boxing
  - An incremental software process is applied to the project
  - The tasks associated with each increment are "time-boxed" (i.e., given a specific start and stop time) by working backward from the delivery date
  - The project is not allowed to get "stuck" on a task
  - When the work on a task hits the stop time of its box, then work ceases on that task and the next task begins
  - This approach succeeds based on the premise that when the time-box boundary is encountered, it is likely that 90% of the work is complete
  - The remaining 10% of the work can be
    - Delayed until the next increment
    - Completed later if required
- Time-boxing is commonly used in agile processes

Agile Process : Burndown Chart



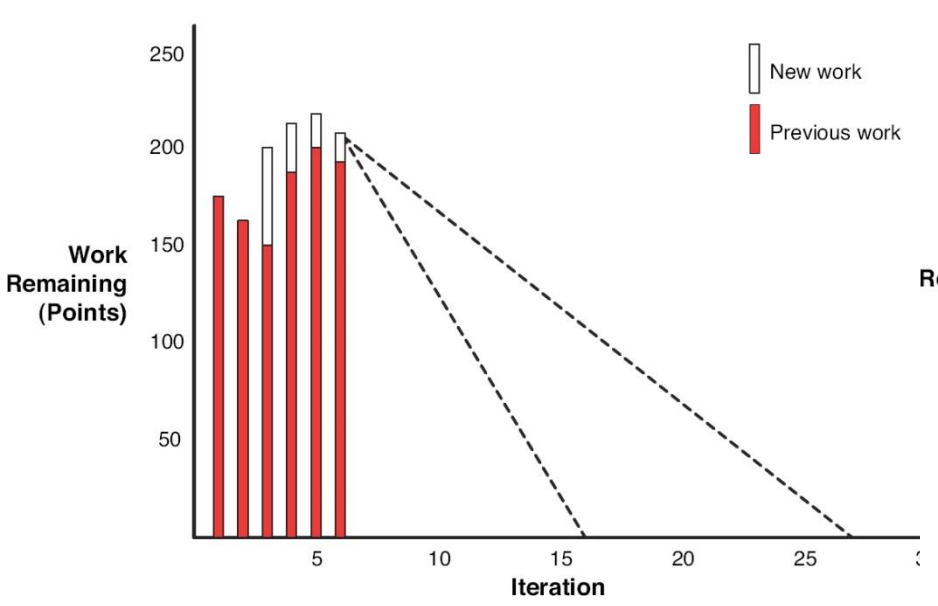
Agile Process Progress

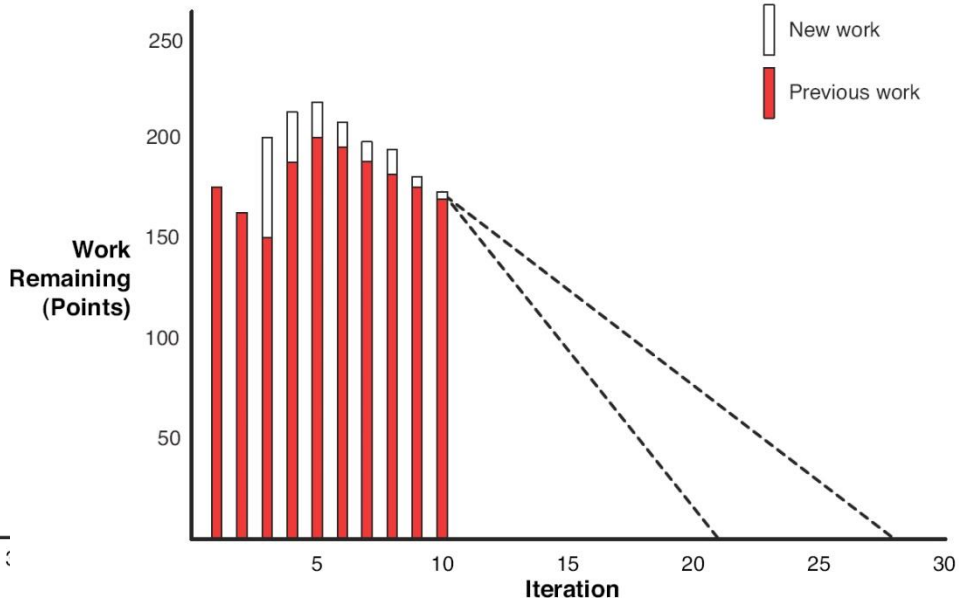


Agile Process : Ranged Estimate

Iteration	Work Remaining	Gross Velocity	Work Added	Net Velocity (actual)	Net Velocity (applied)	Estimated Iterations Remaining	Estimated Schedule
Initial	187	15					13
1	174	13	0	13	13	14	15
2	160	14	0	14	14	12	14
3	200	12	52	"-40"	6	"17-34"	"20-37"
4	213	13	26	"-13"	7	"17-31"	"21-35"
5	218	12	17	"-5"	6	"19-37"	"24-42"
6	208	22	12	10	10	"10-21"	"16-27"
7	198	20	10	10	10	"10-20"	"17-27"
8	195	16	13	3	3	"13-65"	"21-73"
9	182	17	4	13	13	"11-14"	"20-33"
10	172	16	6	10	10	"11-18"	"21-28"
11	159	16	3	13	13	"10-13"	"21-24"
12	144	17	2	15	15	"9-11"	"21-23"
13	129	17	2	15	15	"8-9"	"21-22"

Convergence in Burndown Chart





Gross velocity is the best-case situation and overly optimistic.
Net velocity is the worst-case scenario and overly pessimistic.
Estimates for completion should converge over iterations

Resource Planning

Resource Estimation

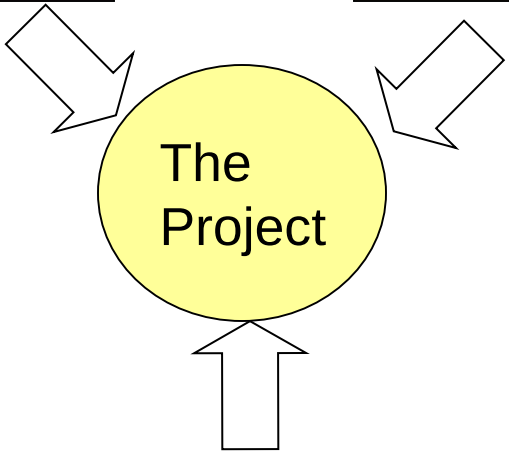
- Three major categories of software engineering resources
  - People
  - Development environment
  - Reusable software components
    - Often neglected during planning but become a paramount concern during the construction phase of the software process
- Each resource is specified with
  - A description of the resource
  - A statement of availability
  - The time when the resource will be required
  - The duration of time that the resource will be applied

Time window

Categories of Resources

People
- Number required
- Skills required
- Geographical location

Development Environment
- Software tools
- Computer hardware
- Network resources



Reusable Software Components
- Off-the-shelf components
- Full-experience components
- Partial-experience components
- New components

Human Resources

- Planners need to select the number and the kind of people skills needed to complete the project
- They need to specify the organizational position and job specialty for each person
- Small projects of a few person-months may only need one individual
- Large projects spanning many person-months or years require the location of the person to be specified also
- The number of people required can be determined only after an estimate of the development effort

Development Environment Resources

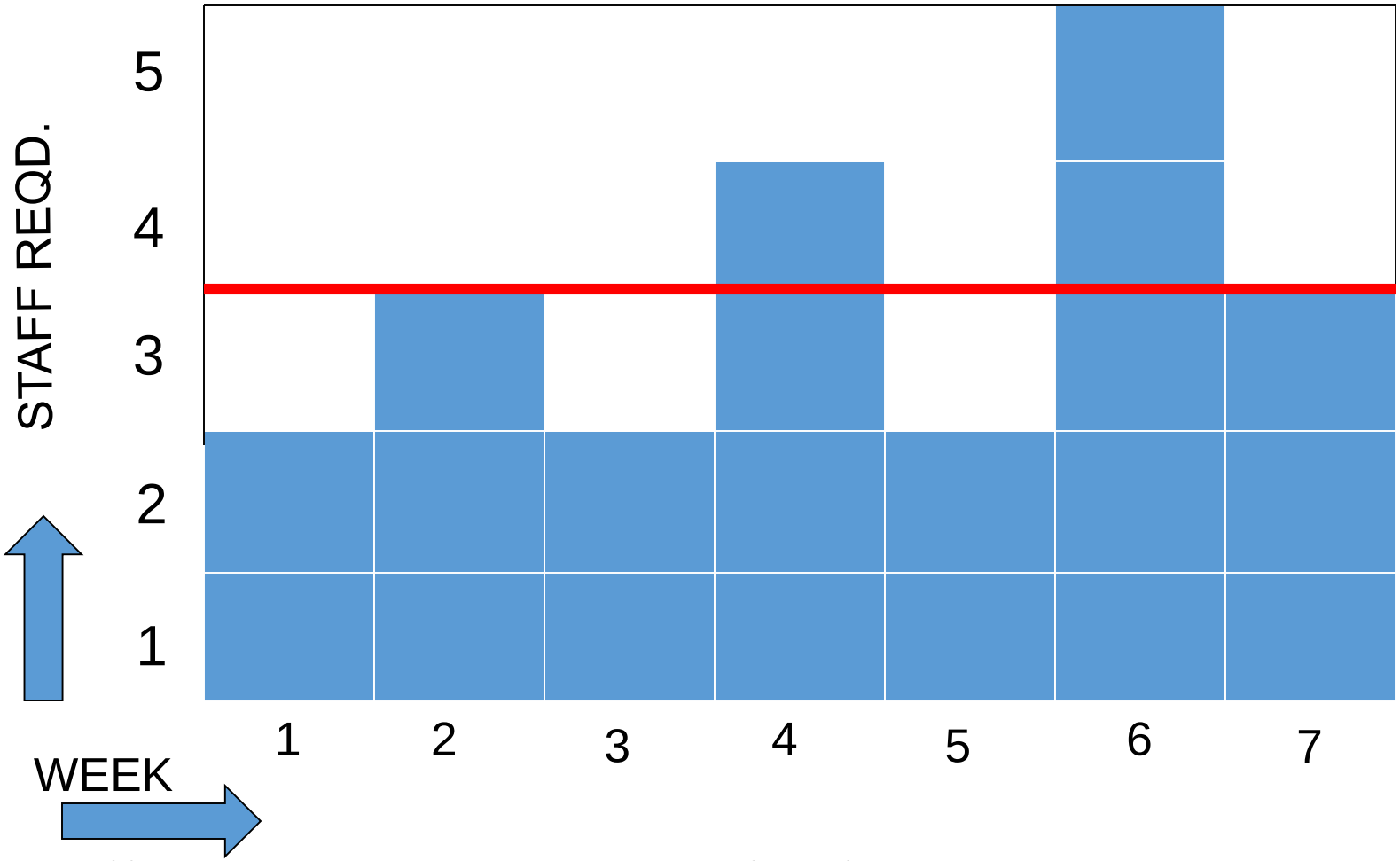
- A software engineering environment (SEE) incorporates hardware, software, and network resources that provide platforms and tools to develop and test software work products
- Most software organizations have many projects that require access to the SEE provided by the organization
- Planners must identify the time window required for hardware and software and verify that these resources will be available

Reusable Software Resources

- Off-the-shelf components
  - Components are from a third party or were developed for a previous project
  - Ready to use; fully validated and documented; virtually no risk
- Full-experience components
  - Components are similar to the software that needs to be built
  - Software team has full experience in the application area of these components
  - Modification of components will incur relatively low risk
- Partial-experience components
  - Components are related somehow to the software that needs to be built but will require substantial modification
  - Software team has only limited experience in the application area of these components
  - Modifications that are required have a fair degree of risk
- New components
  - Components must be built from scratch by the software team specifically for the needs of the current project
  - Software team has no practical experience in the application area
  - Software development of components has a high degree of risk

- E. M. Bennatan

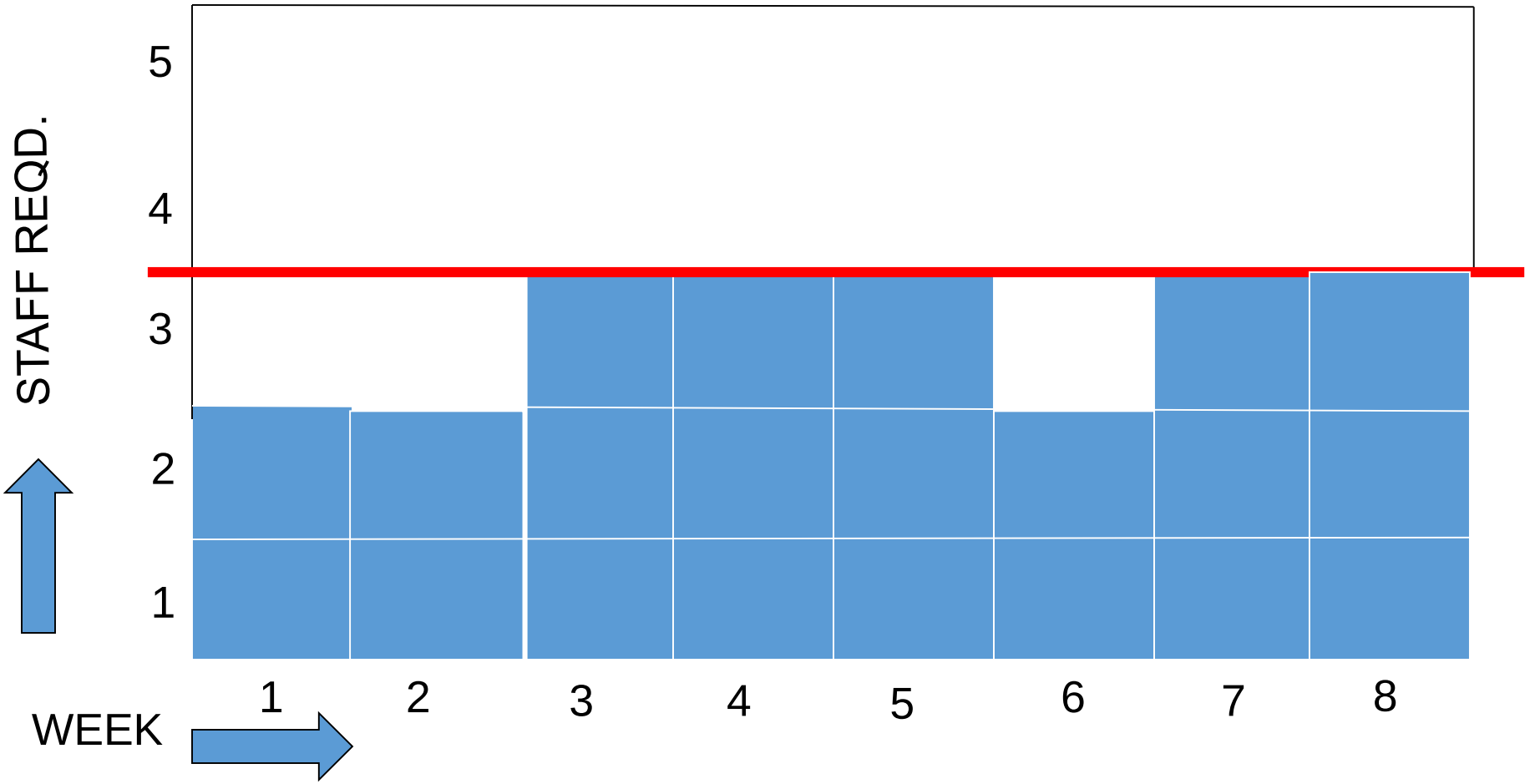
Resource histogram (example)



Resource smoothing

- It is usually difficult to get specialist staff who will work odd days to fill in gaps – need for staff to learn about application etc
- Staff often have to be employed for a continuous block of time
- Therefore desirable to employ a constant number of staff on a project – who as far as possible are fully employed
- Hence need for **resource smoothing**

Resource smoothing



Resource considerations

- Where same resource needed in more than one place at the same time, the conflict can be resolved by:
  - delaying one of the activities
    - taking advantage of float to change start date
    - delaying start of one activity until finish of the other activity that resource is being used on - *puts back project completion*
  - moving resource from a non-critical activity
  - bringing in additional resource - *increases costs*
- At the same time
  - need to maximise %usage of resources i.e. reduce idle periods between tasks
  - need to balance costs against early completion date
  - need to allow for contingency

Prioritizing activities

There are two main ways of doing this:

- *Total float priority* – those with the smallest float have the highest priority
- *Ordered list priority* – this takes account of the duration of the activity as well as the float

Burman's priority list

Give priority to:

- Shortest critical activities
- Other critical activities
- Shortest non-critical activities
- Non-critical activities with least float
- Non-critical activities

Critical path

- Scheduling resources can create new dependencies between activities
- It is best not to add dependencies to the activity network to reflect resource constraints
  - Makes network very messy
  - A resource constraint may disappear during the project, but link remains on network
- Amend dates on **schedule** to reflect resource constraints

Allocating individuals to activities

The initial ‘resource types’ for a task have to be replaced by actual individuals.

Factors to be considered:

- Availability
- Criticality
- Risk
- Training
- Team building – and motivation

Software Project Risks

Definition of Risk

- A risk is a potential problem – it might happen and it might not
- Conceptual definition of risk
  - Risk concerns future happenings
  - Risk involves change in mind, opinion, event, action, place, etc.
  - Risk involves choice and the uncertainty that choice entails
- Two characteristics of risk
  - Uncertainty – the risk may or may not happen, that is, there are no 100% risks (those, instead, are called constraints)
  - Loss – the risk becomes a reality and unwanted consequences or losses occur

Risk Projection

- Risk projection (or estimation) attempts to rate each risk in two ways
  - The probability that the risk is real
  - The consequence of the problems associated with the risk, should it occur
- The project planner, managers, and technical staff perform four risk projection steps (see next slide)
- The intent of these steps is to consider risks in a manner that leads to prioritization
- By prioritizing risks, the software team can allocate limited resources where they will have the most impact

Risk Projection/Estimation Steps

1) Establish a scale that reflects the perceived likelihood of a risk (e.g., 1-low, 10-high)
2) Delineate the consequences of the risk
3) Estimate the impact of the risk on the project and product
4) Note the overall accuracy of the risk projection so that there will be no misunderstandings

Contents of a Risk Table

- A risk table provides a project manager with a simple technique for risk projection
- It consists of five columns
  - Risk Summary – short description of the risk
  - Risk Category – one of seven risk categories (slide 22)
  - Probability – estimation of risk occurrence based on group input
  - Impact – (1) catastrophic (2) critical (3) marginal (4) negligible
  - RMMM – Pointer to a paragraph in the Risk Mitigation, Monitoring, and Management Plan

Risk Summary	Risk Category	Probability	Impact (1-4)	RMMM

(More on next slide)

Developing a Risk Table

- List all risks in the first column (by way of the help of the risk item checklists)
- Mark the category of each risk
- Estimate the probability of each risk occurring
- Assess the impact of each risk based on an averaging of the four risk components to determine an overall impact value (See next slide)
- Sort the rows by probability and impact in descending order
- Draw a horizontal cutoff line in the table that indicates the risks that will be given further attention

Assessing Risk Impact

- Three factors affect the consequences that are likely if a risk does occur
  - **Its nature** – This indicates the problems that are likely if the risk occurs
  - **Its scope** – This combines the severity of the risk (how serious was it) with its overall distribution (how much was affected)
  - **Its timing** – This considers when and for how long the impact will be felt
- The overall risk exposure formula is RE = P x C
  - P = the probability of occurrence for a risk
  - C = the cost to the project should the risk actually occur

Risk Exposure Example

- **Risk identification.** Only 70 percent of the software components scheduled for reuse will, in fact, be integrated into the application. The remaining functionality will have to be custom developed.
- **Risk probability.** 80% (likely).
- **Risk impact.** 60 reusable software components were planned. If only 70 percent can be used, 18 components would have to be developed from scratch (in addition to other custom software that has been scheduled for development). Since the average component is 100 LOC and local data indicate that the software engineering cost for each LOC is $14.00, the overall cost (impact) to develop the components would be 18 x 100 x 14 = $25,200.
- **Risk exposure.** RE = 0.80 x 25,200 ~ $20,200.

Risk types and examples

Risk	Probability	Effects
It is impossible to recruit staff with the skills required for the project.	High	Catastrophic
Organizational financial problems force reductions in the project budget.	Low	Catastrophic
The organization is restructured so that different management are responsible for the project.	High	Serious
The time required to develop the software is underestimated.	High	Serious
Key staff are ill at critical times in the project.	Moderate	Serious
Faults in reusable software components have to be repaired before these components are reused.	Moderate	Serious
Changes to requirements that require major design rework are proposed.	Moderate	Serious

Sommerville, I., Software Engineering, Pearson Education, 9th Ed., 2010

Risk types and examples

Risk	Probability	Effects
The database used in the system cannot process as many transactions per second as expected.	Moderate	Serious
Software tools cannot be integrated.	High	Tolerable
The size of the software is underestimated.	High	Tolerable
Customers fail to understand the impact of requirements changes.	Moderate	Tolerable
Required training for staff is not available.	Moderate	Tolerable
The rate of defect repair is underestimated.	Moderate	Tolerable
Code generated by code generation tools is inefficient.	Moderate	Insignificant

Sommerville, I., Software Engineering, Pearson Education, 9th Ed., 2010

Risk Mitigation, Monitoring, and Management

Background

- An effective strategy for dealing with risk must consider three issues
  (Note: these are not mutually exclusive)
- **mitigation**—how can we avoid the risk?
- **monitoring**—what factors can we track that will enable us to determine if the risk is becoming more or less likely?
- **management**—what contingency plans do we have if the risk becomes a reality?

- Risk mitigation (avoidance) is the primary strategy and is achieved through a plan
  - Example: Risk of high staff turnover (see next slide)

(More on next slide)

Background (continued)

- During risk monitoring, the project manager monitors factors that may provide an indication of whether a risk is becoming more or less likely
- Risk management and contingency planning assume that mitigation efforts have failed and that the risk has become a reality
- RMMM steps incur additional project cost
  - Large projects may have identified 30 – 40 risks
- Risk is not limited to the software project itself
  - Risks can occur after the software has been delivered to the user

(More on next slide)

Background (continued)

- Software safety and hazard analysis
  - These are software quality assurance activities that focus on the identification and assessment of potential hazards that may affect software negatively and cause an entire system to fail
  - If hazards can be identified early in the software process, software design features can be specified that will either eliminate or control potential hazards

Background (continued)

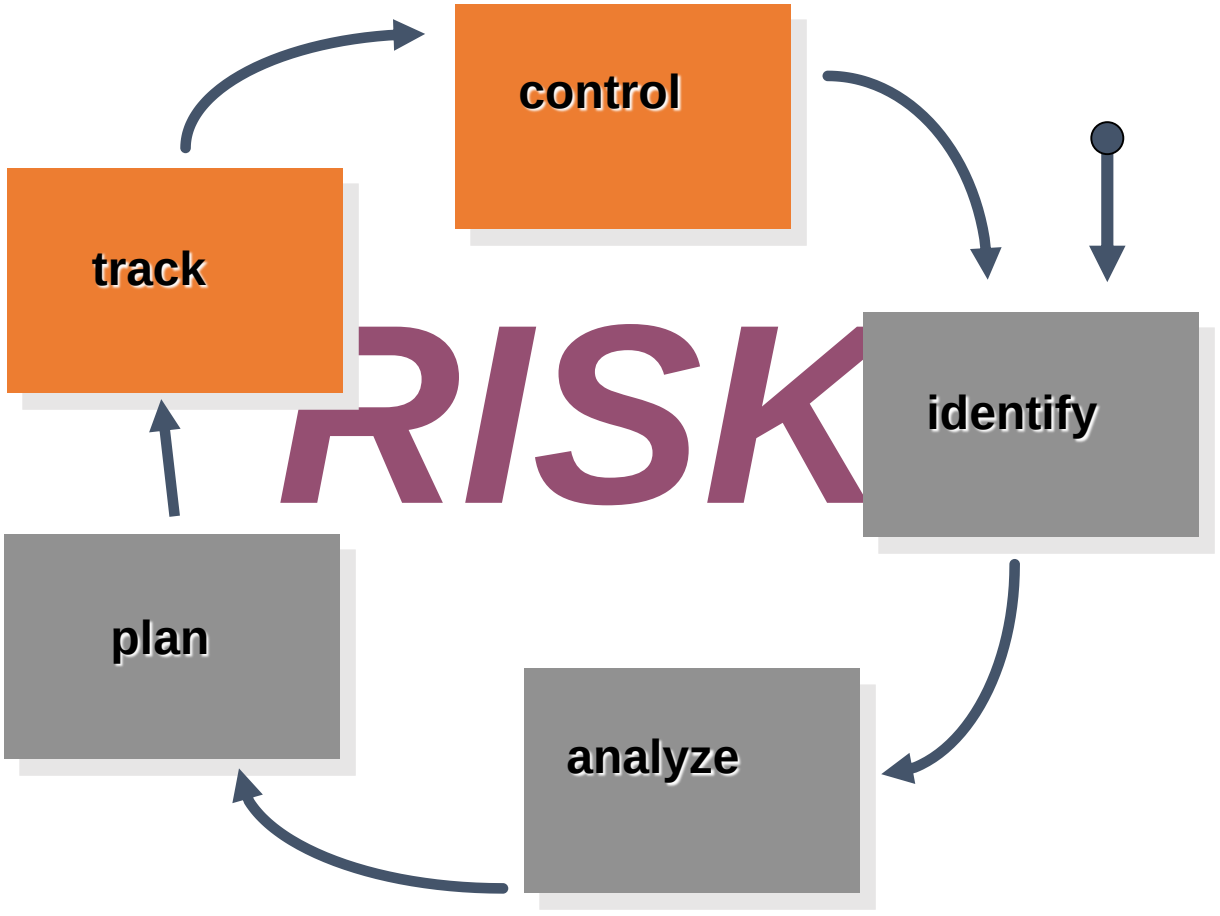
Strategy for Reducing Staff Turnover

- ❑Meet with current staff to determine causes for turnover (e.g., poor working conditions, low pay, competitive job market)
- ❑Mitigate those causes that are under our control before the project starts
- ❑Once the project commences, assume turnover will occur and develop techniques to ensure continuity when people leave
- ❑Organize project teams so that information about each development activity is widely dispersed
- ❑Define documentation standards and establish mechanisms to ensure that documents are developed in a timely manner
- ❑Conduct peer reviews of all work (so that more than one person is "up to speed")
- ❑Assign a backup staff member for every critical technologist

The RMMM Plan

- The RMMM plan may be a part of the software development plan or may be a separate document
- Once RMMM has been documented and the project has begun, the risk mitigation, and monitoring steps begin
  - Risk mitigation is a problem avoidance activity
  - Risk monitoring is a project tracking activity
- Risk monitoring has three objectives
  - To assess whether predicted risks do, in fact, occur
  - To ensure that risk aversion steps defined for the risk are being properly applied
  - To collect information that can be used for future risk analysis
- The findings from risk monitoring may allow the project manager to ascertain what risks caused which problems throughout the project

Risk Management Paradigm



Seven Principles of Risk Management

- **Maintain a global perspective**
  - View software risks within the context of a system and the business problem that is is intended to solve
- **Take a forward-looking view**
  - Think about risks that may arise in the future; establish contingency plans
- **Encourage open communication**
  - Encourage all stakeholders and users to point out risks at any time
- **Integrate risk management**
  - Integrate the consideration of risk into the software process
- **Emphasize a continuous process of risk management**
  - Modify identified risks as more becomes known and add new risks as better insight is achieved
- **Develop a shared product vision**
  - A shared vision by all stakeholders facilitates better risk identification and assessment
- **Encourage teamwork when managing risk**
  - Pool the skills and experience of all stakeholders when conducting risk management activities

- SEI (Software Engineering Institute)

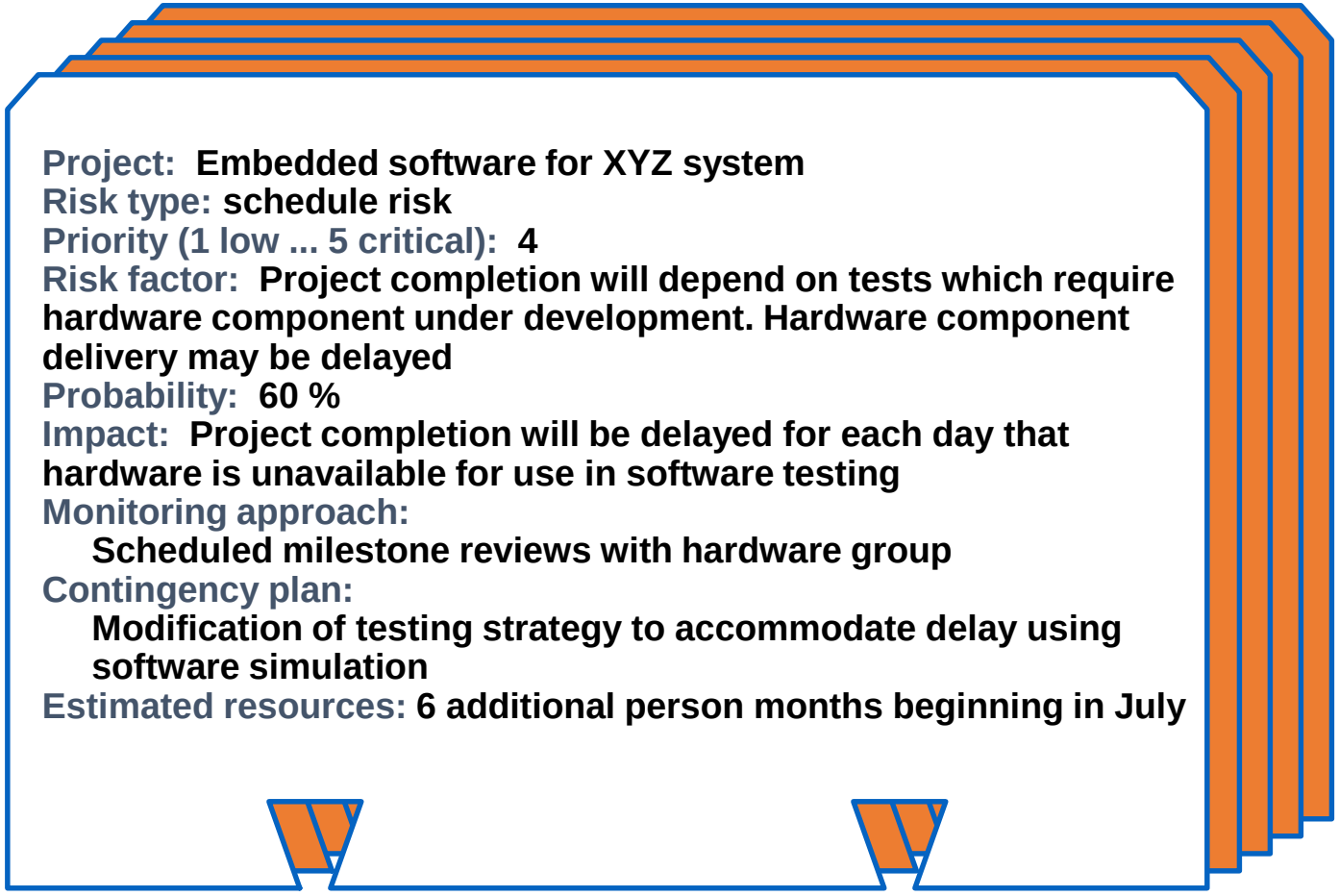
Strategies to help manage risk

Risk	Strategy
Organizational financial problems	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business and presenting reasons why cuts to the project budget would not be cost-effective.
Recruitment problems	Alert customer to potential difficulties and the possibility of delays; investigate buying-in components.
Staff illness	Reorganize team so that there is more overlap of work and people therefore understand each other's jobs.
Defective components	Replace potentially defective components with bought-in components of known reliability.
Requirements changes	Derive traceability information to assess requirements change impact; maximize information hiding in the design.

Strategies to help manage risk

Risk	Strategy
Organizational restructuring	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business.
Database performance	Investigate the possibility of buying a higher-performance database.
Underestimated development time	Investigate buying-in components; investigate use of a program generator.

Recording Risk Information



Risk Information Sheet (RIS), proposed by Williams, Walker, Dorofee, is a complete documentation for each risk. RIS is preferred by some practitioners over RMMM.

Thank You

Any Questions?