Module 4: Software Project Management (13 hrs) – Application level

Project Management Concepts, Project Planning, Overview of metrics, Estimation for Software projects, Project Scheduling, Risk Management, Maintenance and Reengineering, Software Process Improvement (SPI): CMM Levels.

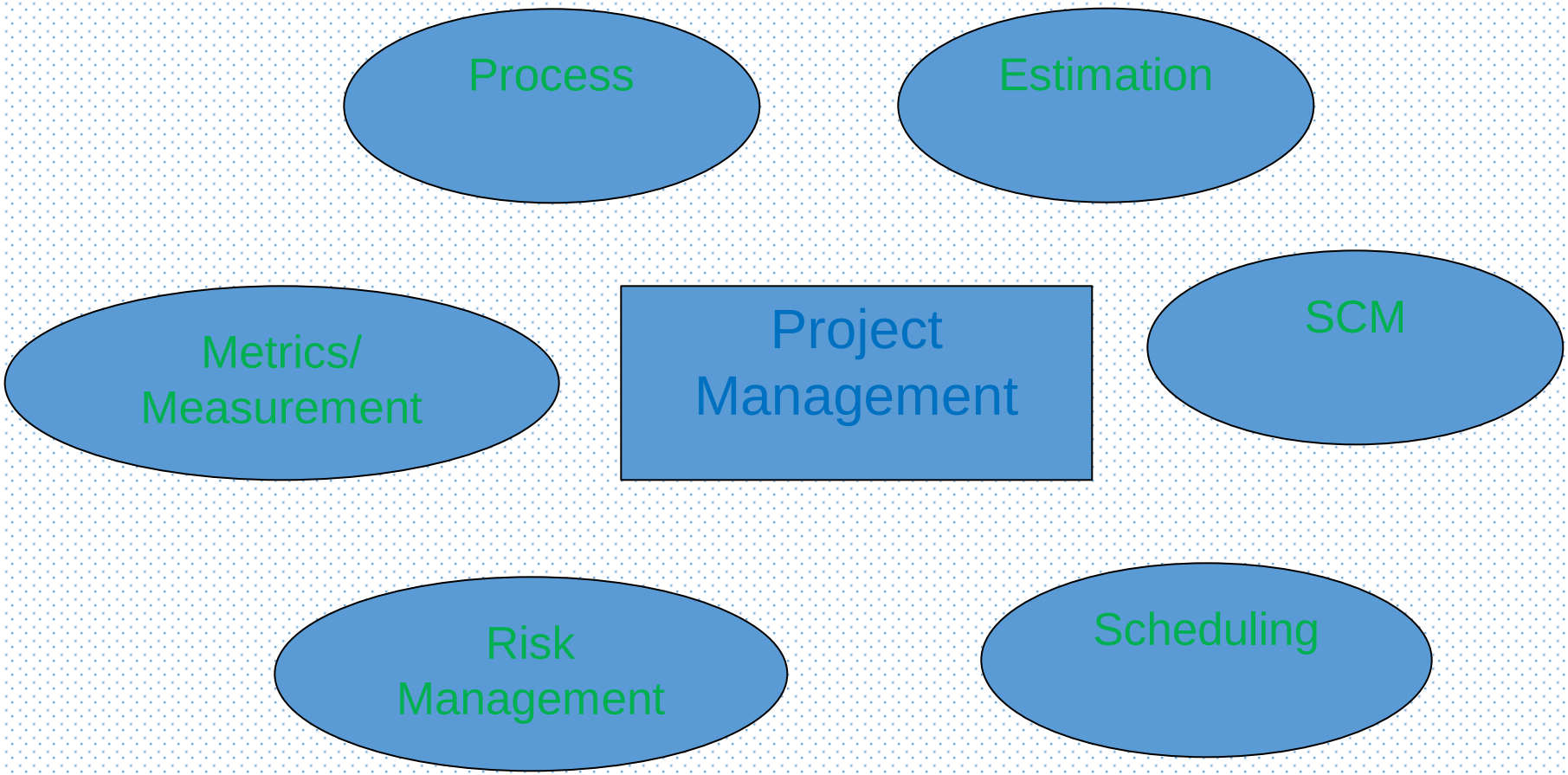
Software Project Management

Department of Computer Science and Engineering
School of Engineering, Presidency University

What is Software Project Management

Software project management is an **art and science of planning and leading software projects**. It is a sub-discipline of project management in which software projects are **planned, implemented, monitored** and **controlled**.

What is Software Project Management



4Ps of Software Project Management

- 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

- ***The following factors must be considered when selecting a software project team structure ...***
  - the difficulty of the problem to be solved
  - the size of the resultant program(s) in lines of code or function points
  - the time that the team will stay together (team lifetime)
  - the degree to which the problem can be modularized
  - the required quality and reliability of the system to be built
  - the rigidity of the delivery date
  - the degree of sociability (communication) required for the project

Organizational Paradigms
How teams are formed

- Closed paradigm - structures a team along a traditional hierarchy of authority
- Random paradigm - structures a team loosely and depends on individual initiative of the team members
- 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
- 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

Team Coordination and Communication

- *Formal, impersonal approaches* include software engineering documents and work products (including source code), technical memos, project milestones, schedules, and project control tools, change requests and related documentation, error tracking reports, and repository data.
- *Formal, interpersonal procedures* focus on quality assurance activities 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 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

- *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.
- *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.
- *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.
- *Make smart decisions.* In essence, the decisions of the project manager and the software team should be to "keep it simple."
- *Conduct a postmortem analysis.* Establish a consistent mechanism for extracting lessons learned for each project.

Understanding Essence of Project

- Why is the system being developed?
- What will be done?
- When will it be accomplished?
- Who is responsible?
- Where are they organizationally located?
- 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

Module 4: Software Project Management (13 hrs) – Application level

Project Management Concepts, Project Planning, Overview of metrics, Estimation for Software projects, Project Scheduling, Risk Management, Maintenance and Reengineering, Software Process Improvement (SPI): CMM Levels.

Project Planning

Department of Computer Science and Engineering
School of Engineering, Presidency University

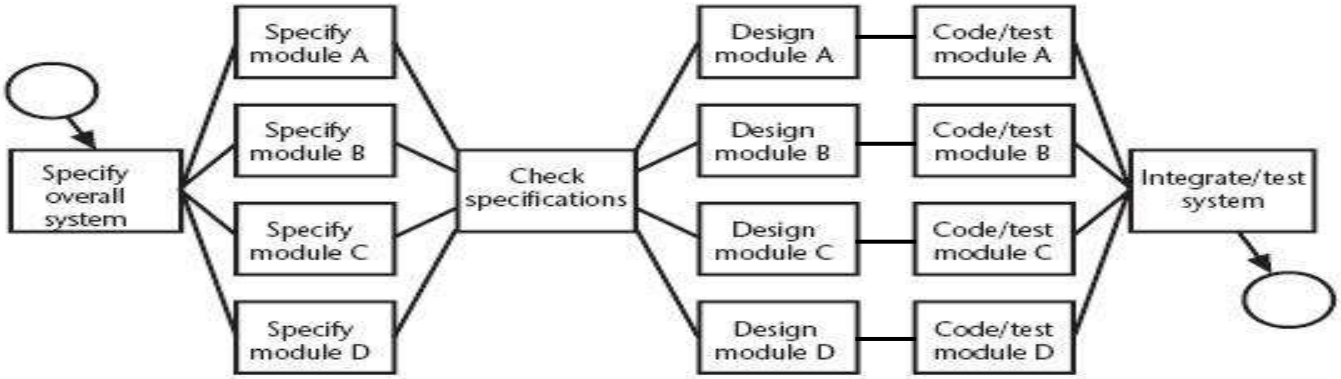
Project Activity Planning

- Project Activity Planning happens before the Scheduling phase.
- Here, activities (tasks) are identified.
- "**Network Planning Models**" are created that sequence the tasks based on logical relationship, and subsequently schedule based on the resource constraints. In this, the project's activities and their relationships are modelled as a **network**, in which time flows from left to right.

Network Planning Model: **Activity-on-Arrow**

- Used to visualize the project as a network
- Activities are drawn as arrow joining circles, or nodes, which represent the possible start and/or completion of an activity or set of activities

Sample Fragment of a network developed as activity-on-node



Formulating a Network Model

Constructing Precedence Network Rules-I

- A project network should have only one start node
  - More than one activity starting at once? Invent a ‘start’ activity with zero duration
- A project network should have only one end node
  - If necessary, invent an ‘end’ activity
- A node has duration
- Links normally have no duration

Formulating a Network Model

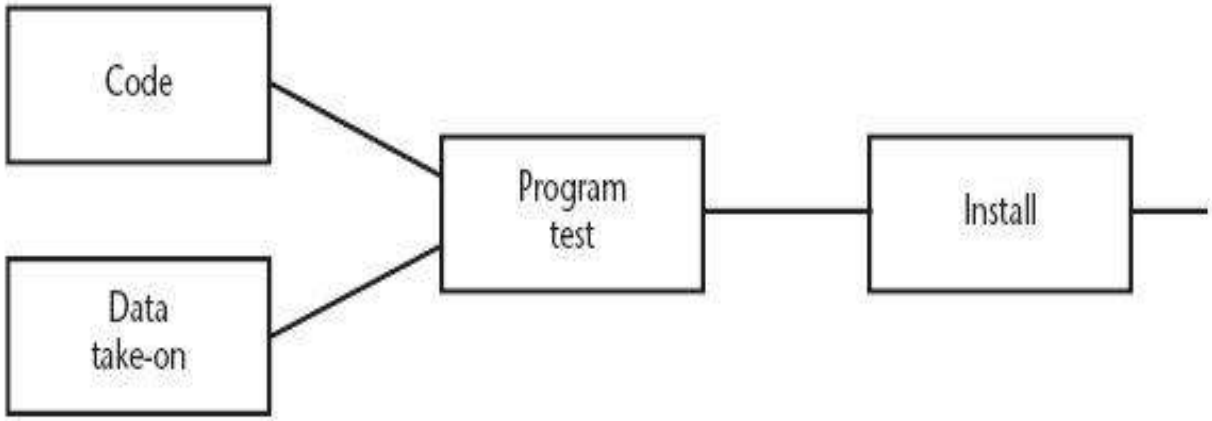
Constructing Precedence Network Rules-II

- Precedents are the immediate preceding activities
  - All have to be completed before an activity can be started
- Time moves from left to right
- A network may not contain loops
- A network should not contain dangles
  - If necessary, connect to the final node

Formulating a Network Model

Fragment of Precedence Network

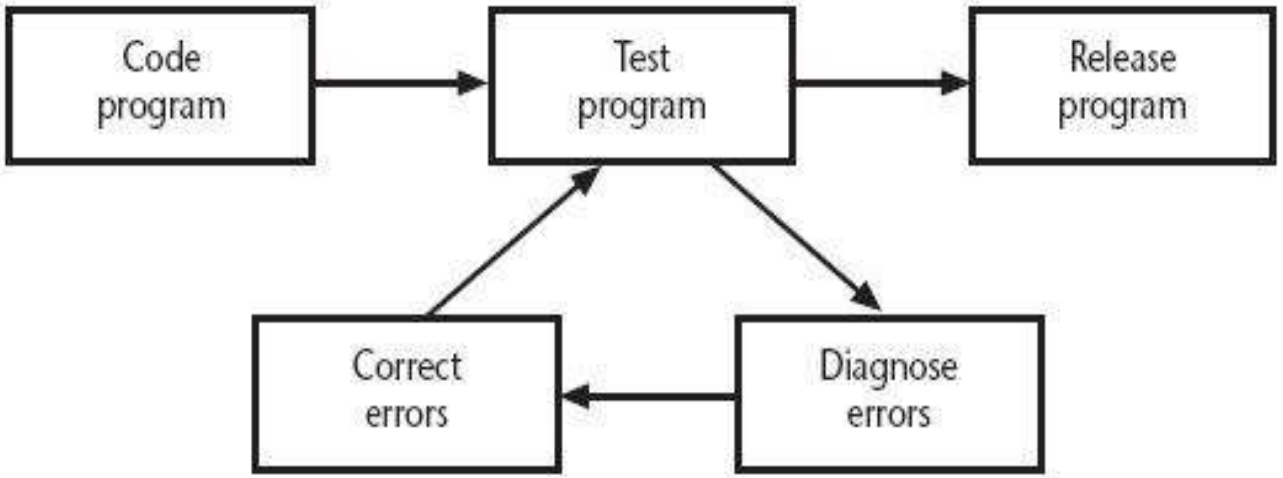
- Installation cannot start until program testing is completed
- Program test cannot start until both code and data take-on have been completed



Formulating a Network Model

Network contains Loop

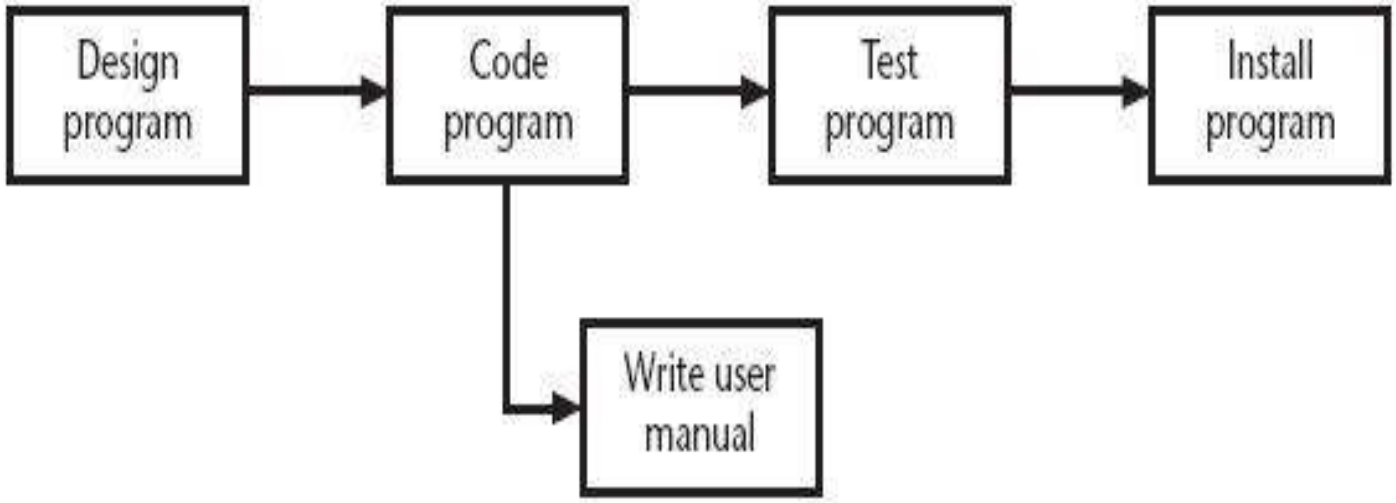
- A loop is an error in that it represents a situation that cannot occur in practice
  - Program testing cannot start until errors have been corrected?



Formulating a Network Model

A Dangle

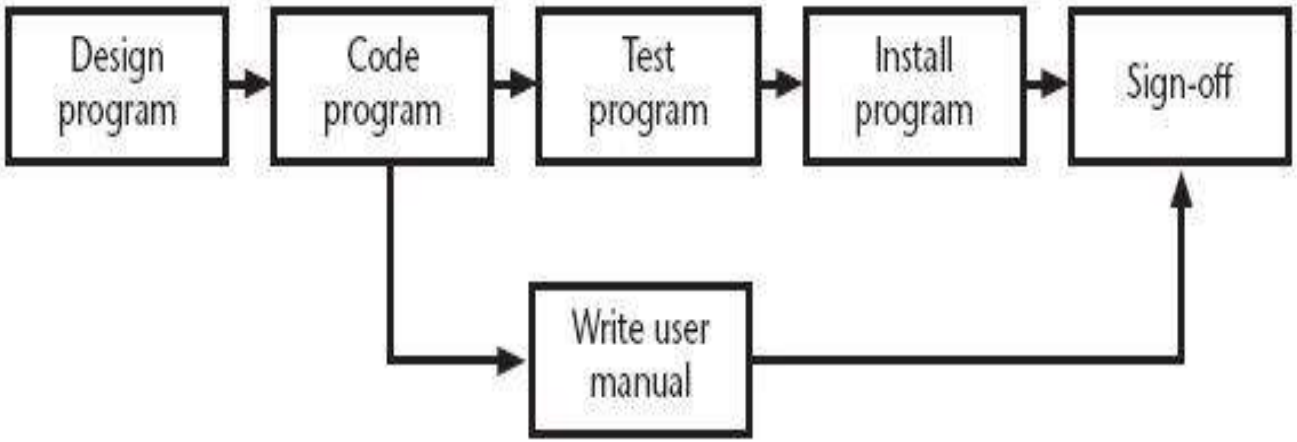
- A dangling activity such as “write user manual” should not exist as it is likely to lead to errors in subsequent analysis



Formulating a Network Model

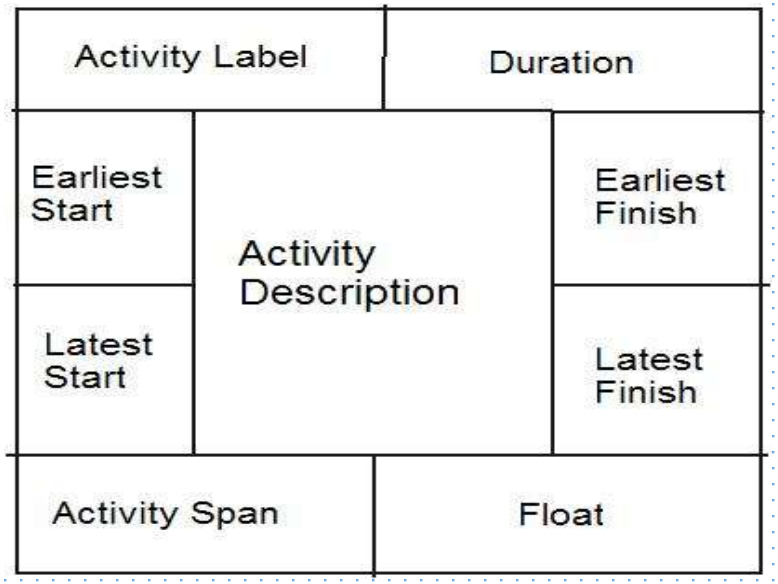
Resolving the Dangle

- The figure implies that the project is complete once the software has been installed and the user manual written
  - We should redraw the network with a final completion activity



Formulating a Network Model

Labelling Convention



Critical Path Approach

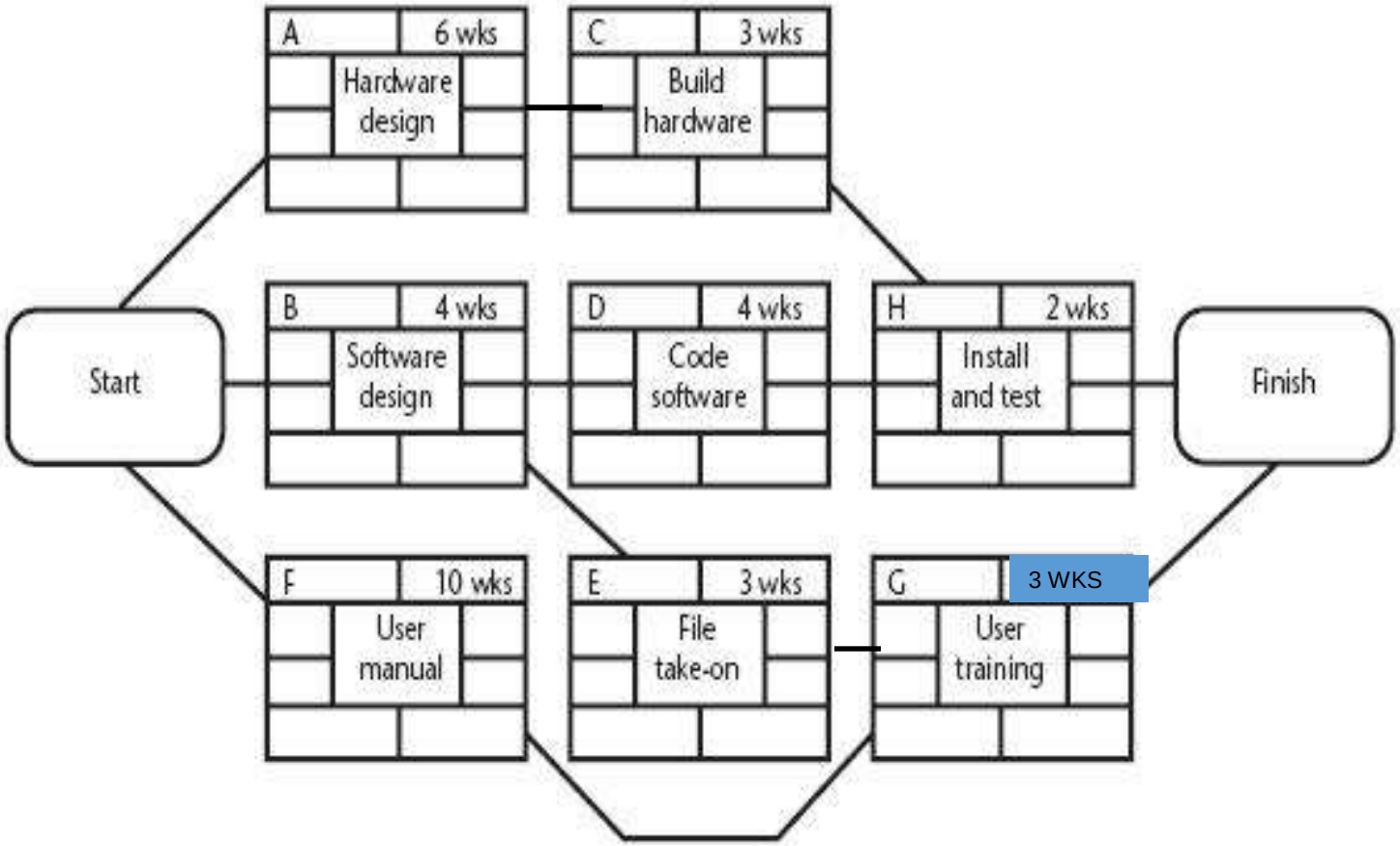
- Planning the project in such way that it is completed as quickly as possible
- Identifying delayed activities

- The method requires the estimation of duration of each activity
  - ***Forward pass***: Calculate the earliest dates at which activities may commence and the project completed
  - ***Backward pass***: Calculate the latest start dates for activities and the *critical path*

Estimated Activity Duration of A Project

Activity		*Duration (weeks)*	*Precedents*
A	Hardware selection	6	
B	Software design	4	
C	Install hardware	3	A
D	Code & test software	4	B
E	File take-on	3	B
F	Write user manuals	10	
G	User training	3	E,F
H	Install & test system	2	C,D

Precedence Network for the Project



The Forward Pass
Calculation of Earliest Start Date - I

- Activities A, B and F may start immediately
  - The earliest date for their start is zero
- Activity A will take 6 weeks
  - The earliest it can finish is week 6
- Activity F will take 10 weeks
  - The earliest it can finish is week 10

The Forward Pass
Calculation of Earliest Start Date - II

- Activity C can start as soon as A has finished
  - Its earliest start date is week 6
  - It will take 3 weeks, so the earliest it can finish is week 9
- Activities D and E can start as soon as B is complete
  - The earliest they can each start is week 4
  - Activity D will take 4 weeks, so the earliest it can finish is week 8
  - Activity E will take 3 weeks, so the earliest it can finish is week 7

The Forward Pass
Calculation of Earliest Start Date - III

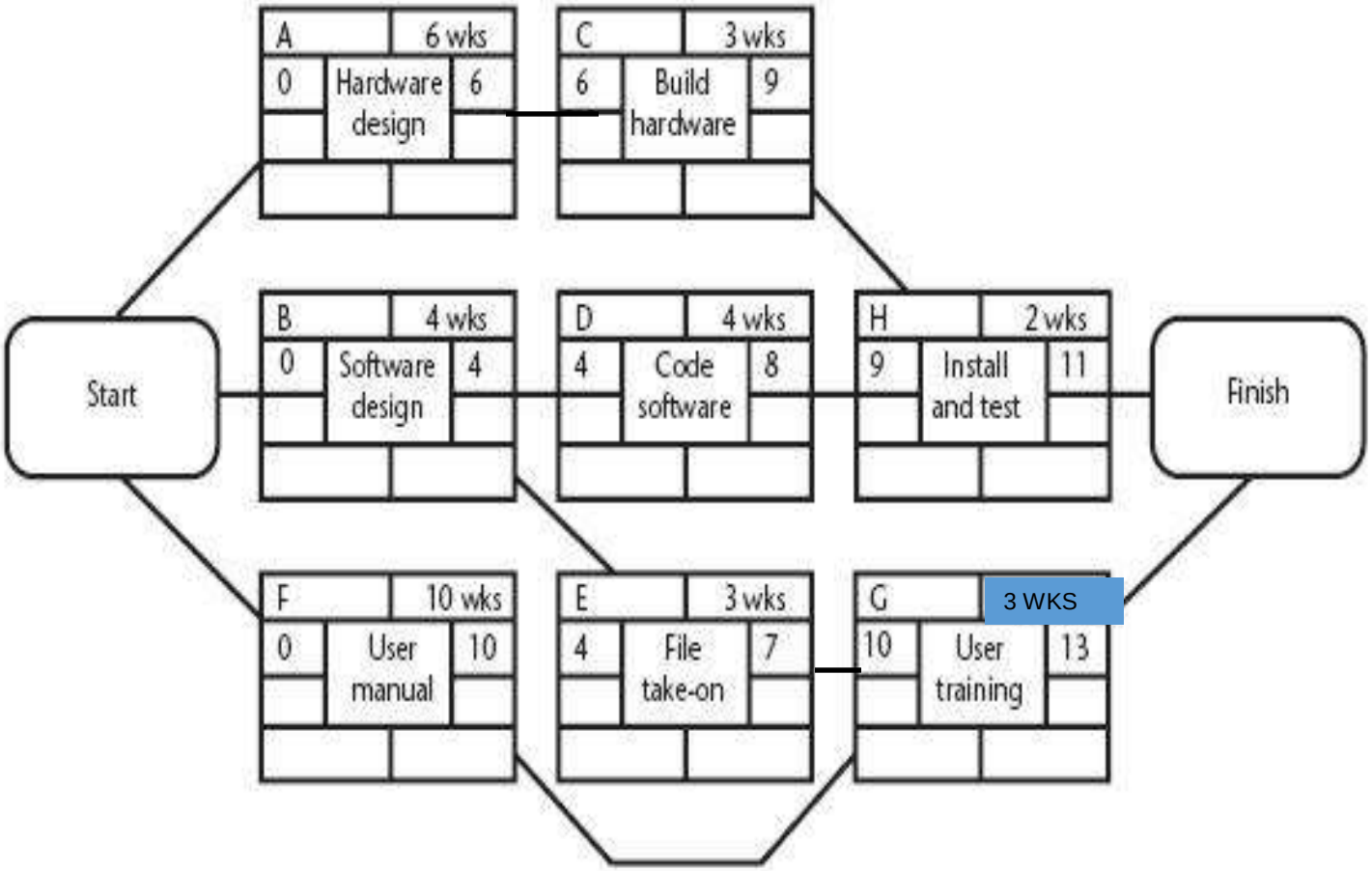
- Activity G cannot start until both E and F have been completed
  - It cannot start until week 10 - the later of weeks 7 (activity E) and 10 (for activity F)
  - It takes 3 weeks and finishes in week 13
- Similarly, activity H cannot start until week 9 – the later of the two earliest finished dates for the preceding activities C and D

The Forward Pass
Calculation of Earliest Start Date - IV

- The project will be complete when both activities H and G have been completed
  - The earliest project completion date will the later of weeks 11 and 13 – that is, week 13

The Forward Pass

The Network after the Forward Pass



The Backward Pass

The Latest Activity Dates Calculation-I

- The latest completion date for activities G and H is assumed to be week 13
- Activity H must therefore start at week 11 at the latest (13-2) and the latest start date for activity G is week 10 (13-3)
- The latest completion date for activities C and D is the latest date at which activity H must start – that is week 11
  - The latest start date of week 8 (11-3), and week 7 (10-3) respectively

The Backward Pass

The Latest Activity Dates Calculation-II

- Activities E and F must be completed by week 10
  - The earliest start dates are weeks 7 (10-3) and 0 (10-10) respectively
- Activity B must be completed by week 7 (the latest start date for both activities D and E)
  - The latest start is week 3 (7-4)

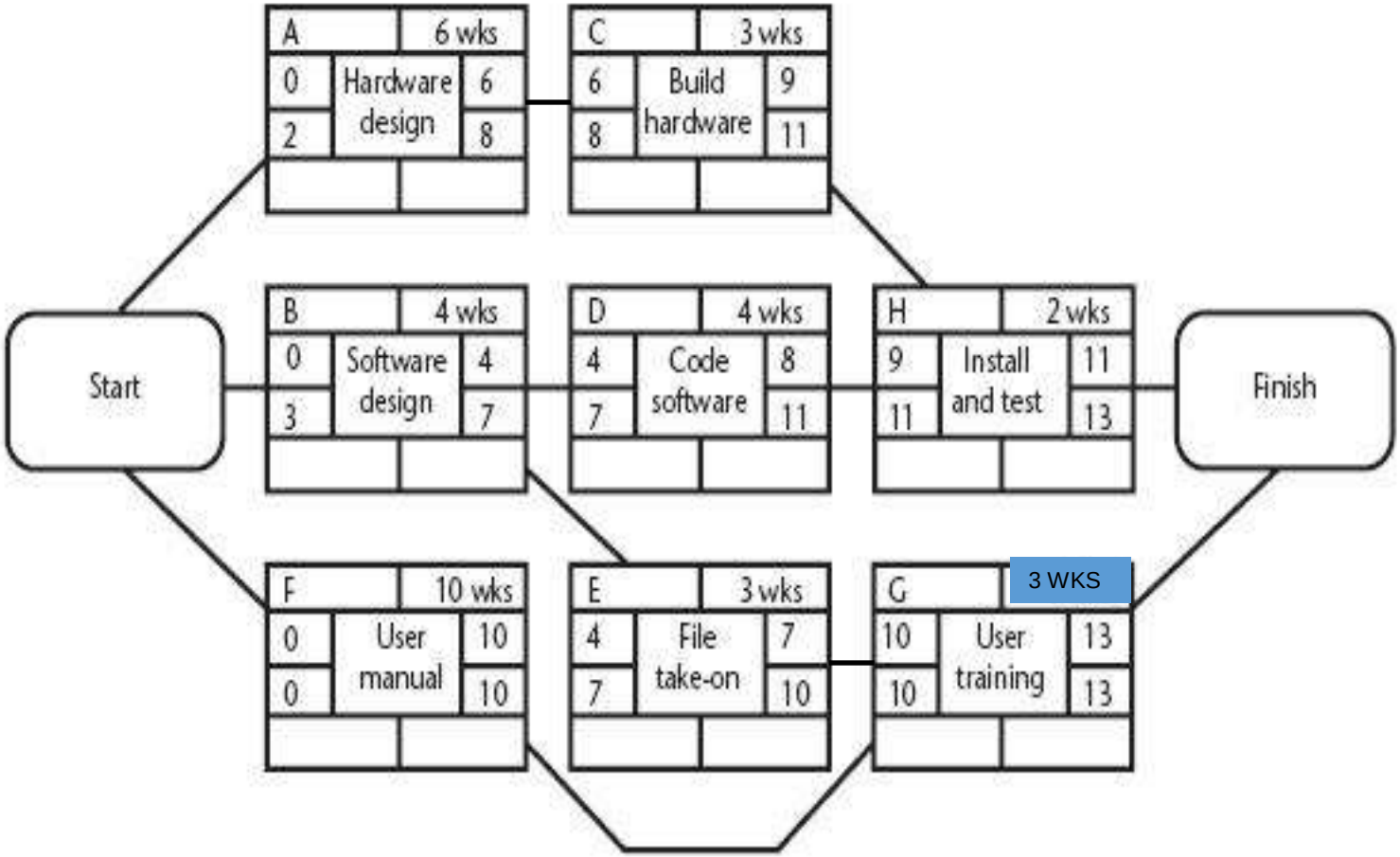
The Backward Pass

The Latest Activity Dates Calculation-III

- Activity A must be completed by week 8 (the latest start date for activity C)
  - Its latest start is week 2 (8-6)
- The latest start date for the project start is the earliest of the latest start dates for activities A, B and F
  - This week is week zero
  - It tells us that if the project does not start on time it won't finish on time

The Backward Pass

The Network after the Backward Pass



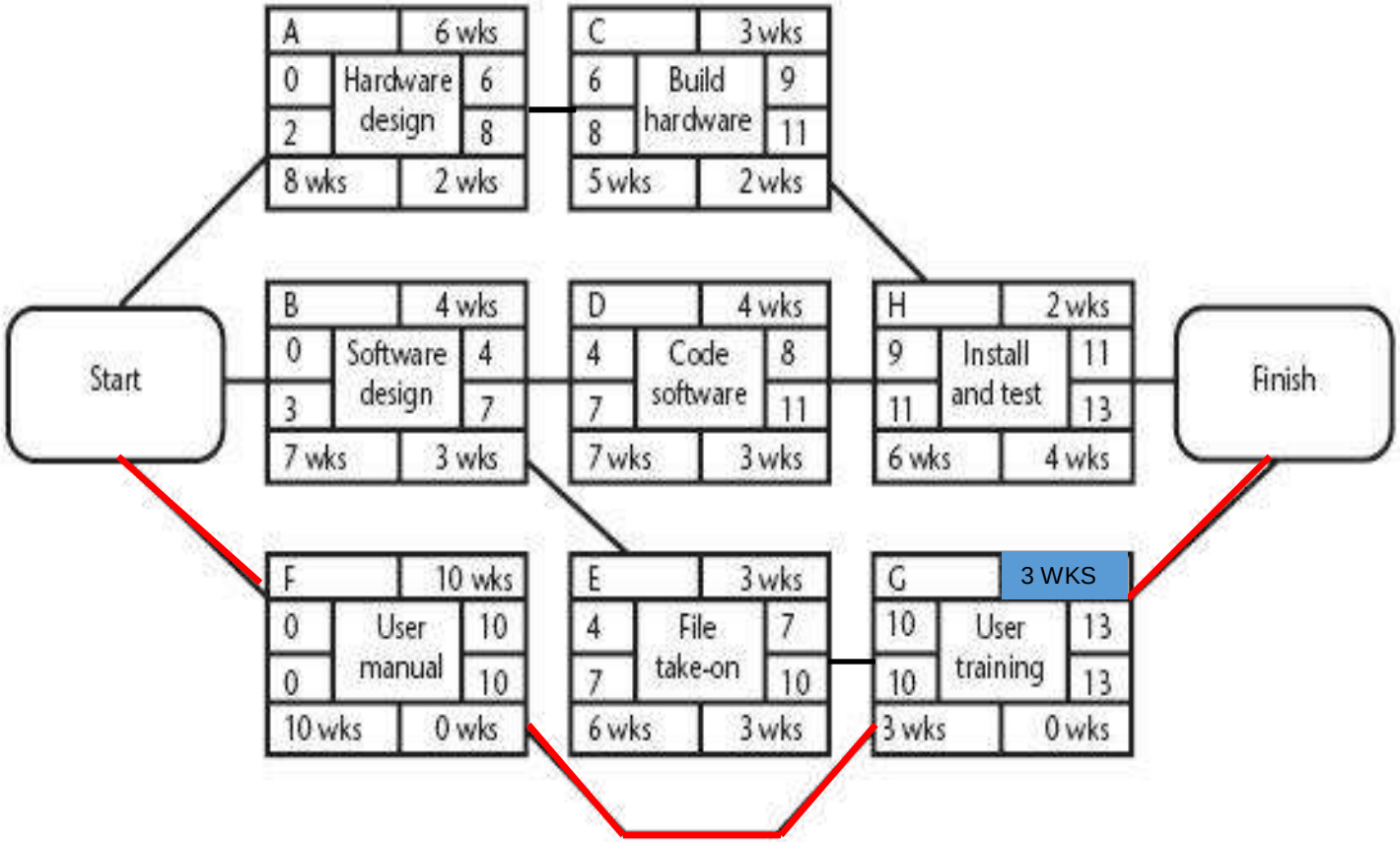
Identifying the Critical Path

- ***Critical path*:** One path through the network that defines the duration of the project
- Any delay to any activity of this critical path will delay the completion of the project

Identifying Float and Span

- **Activity's *float***: the difference between an activity's earliest start date and its latest start date (or, equally, the difference between its earliest and latest finish dates)
  - A measure of how much the start date or completion of an activity may be delayed without affecting the end date of the project
- ***Activity span*:** the difference between the earliest start date and the latest finish date
  - Measure of maximum time allowable for the activity

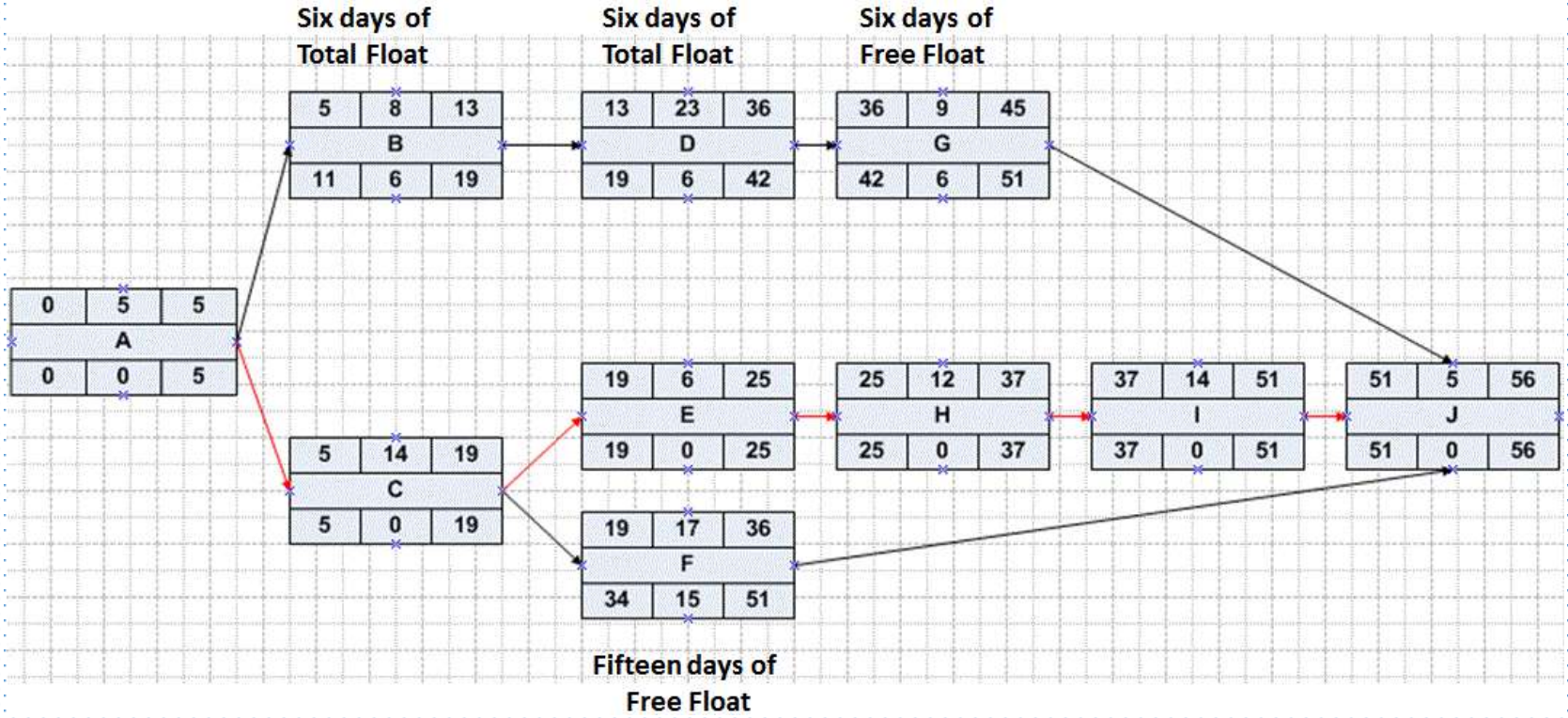
The Critical Path for the Example



Significance of the Critical Path

- In managing the project, we must pay particular attention to monitoring activities on the critical path
  - The effects on any delay or resources unavailability are detected and corrected at the earliest opportunity
- In planning project, it is the critical path that we must shorten if we are to reduce the overall duration of the project

Activity	Duration	Predecessor
A	5	-
B	8	A
C	14	A
D	23	B
E	6	C
F	17	C
G	9	D
H	12	E
I	14	H
J	5	F,G,I


Six days of
Total Float
Six days of
Total Float
Six days of
Free Float
5 8 13
B
11 6 19
13 23 36
D
19 6 42
36 9 45
G
42 6 51
0 5 5
A
0 0 5
5 14 19
C
5 0 19
19 6 25
E
19 0 25
25 12 37
H
25 0 37
37 14 51
I
37 0 51
51 5 56
J
51 0 56
19 17 36
F
34 15 51
Fifteen days of
Free Float

Module 4: Software Project Management (13 hrs) – Application level

Project Management Concepts, Project Planning, Overview of metrics, Estimation for Software projects, Project Scheduling, Risk Management, Maintenance and Reengineering, Software Process Improvement (SPI): CMM Levels.

Product Metrics

Department of Computer Science and Engineering
School of Engineering, Presidency University

Software Metric

A software **metric** is a standard of measure of a degree to which a software system or process possesses some property.

Metrics can be defined for requirements, design, coding, testing stages of SDLC.

Function Point (FP) Metric

The *function point (FP) metric* can be used effectively as a means for measuring the functionality delivered by a system.

A **Function Point** (FP) is a unit of measurement to express the amount of business functionality, an information system (as a product) provides to a user. FPs measure **software size.** They are widely accepted as an industry standard for functional sizing.

Function Point (FP) Metric

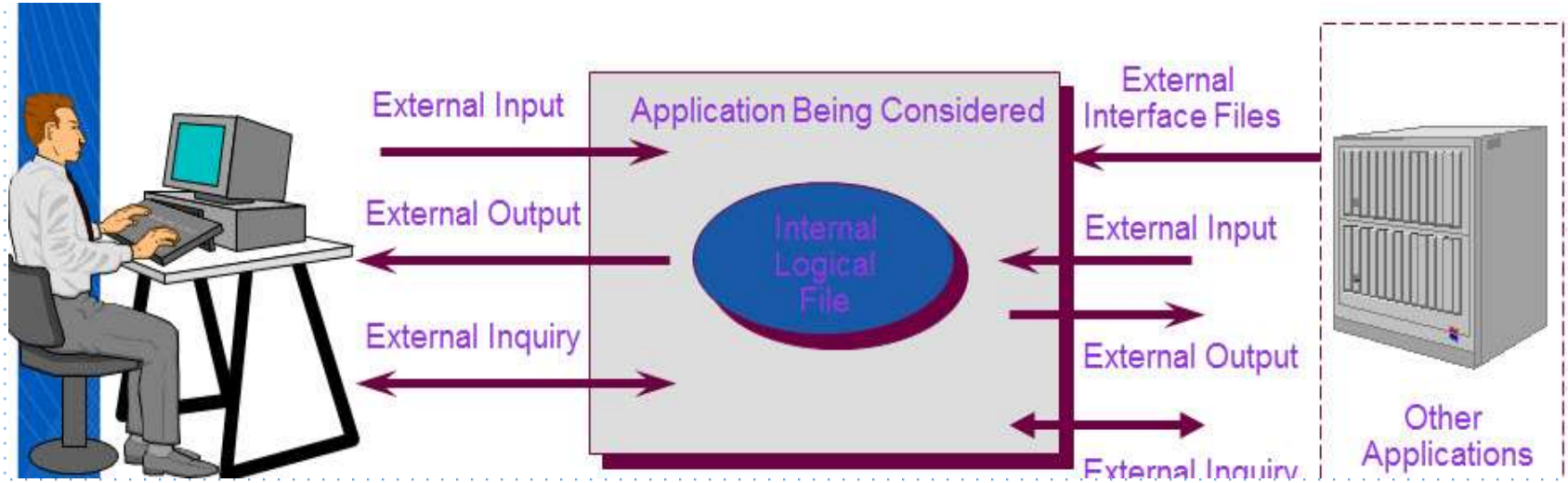
The FP metric can be used to:

(1) estimate the cost or effort required to design, code, and test the software;

(2) predict the number of errors that will be encountered during testing; and

(3) forecast the number of components and/or the number of projected source lines in the implemented system.

Function Point (FP) Metric



Information Domain Values:

1. Number of External Inputs
2. Number of External Outputs
3. Number of External Inquiries
4. Number of Internal Logical Files
5. Number of External Interface Files

FP - Information Domain Values

Number of External Inputs

- Each external input originates from a user or is transmitted from another application
- They provide distinct application-oriented data or control information
- They are often used to update internal logical files
- They are not inquiries (those are counted under another category)

FP - Information Domain Values

Number of External Outputs

- Each external output is derived within the application and provides information to the user
- This refers to reports, screens, error messages, etc.
- Individual data items within a report or screen are not counted separately

FP - Information Domain Values

Number of External Inquiries

- An external inquiry is defined as an online input that results in the generation of some immediate software response
- The response is in the form of an on-line output.

Number of Internal Logical Files

- Each internal logical file is a logical grouping of data that resides within the application's boundary and is maintained via external inputs

FP - Information Domain Values

Number of External Interface Files

- Each external interface file is a logical grouping of data that resides external to the application but provides data that may be of use to the application

Compute the number of function points (FP)

$$FP = count\ total * [0.65 + 0.01 * sum(F_i)]$$

Function Point Calculator:

http://groups.umd.umich.edu/cis/course.des/cis375/projects/fp99/fp_leader.html

Fi – Value Adjustment Factors

The *Fi (i 1 to 14) are value adjustment factors (VAF) based on responses to the* following questions:

1. Does the system require reliable backup and recovery?
2. Are specialized data communications required to transfer information to or from the application?
3. Are there distributed processing functions?
4. Is performance critical?
5. Will the system run in an existing, heavily utilized operational environment?
6. Does the system require online data entry?
7. Does the online data entry require the input transaction to be built over multiple screens or operations?

Fi – Value Adjustment Factors

8. Are the ILFs updated online?
9. Are the inputs, outputs, files, or inquiries complex?
10. Is the internal processing complex?
11. Is the code designed to be reusable?
12. Are conversion and installation included in the design?
13. Is the system designed for multiple installations in different organizations?
14. Is the application designed to facilitate change and ease of use by the user?

Computing Function Points

Function Point Metrics

Parameter	Count		Simple	Average	Complex	
Inputs		x	3	4	6	=
Outputs		x	4	5	7	=
Inquiries		x	3	4	6	=
Files		x	7	10	15	=
Interfaces		x	5	7	10	=
					Count-total (raw FP)	

Function Point Example

Information Domain Value	Count		Weighting Factor: Simple	Average	Complex	
External Inputs	3	x	3	4	6	= 9
External Outputs	2	x	4	5	7	= 8
External Inquiries	2	x	3	4	6	= 6
Internal Logical Files	1	x	7	10	15	= 7
External Interface Files	4	x	5	7	10	= 20
Count total						**50**

- FP = count total * [0.65 + 0.01 * sum(F_i)]
- FP = 50 * [0.65 + (0.01 * 46)]
- FP = 55.5 (rounded up to 56)

Productivity = FP/ Person_month

Quality = Defects / FP

Architectural Design Metrics

These metrics place emphasis on the architectural structure and effectiveness of modules or components within the architecture.

They are "black box" in that they do not require any knowledge of the inner workings of a particular software component.

Card and Glass define three software design complexity measures:

A. Structural complexity
B. Data complexity
C. System complexity

Architectural Design Metrics

For hierarchical architectures (e.g., call-and-return architectures), **structural complexity of a module i** is defined in the following manner:

S(i) = fout(i)

- where fout(i) is the fan-out of module i.
- Fan-out is defined as the number of modules immediately subordinate to module i; that is, the number of modules that are directly invoked by module i.

Architectural Design Metrics

Data complexity provides an indication of the complexity in the internal interface for a module i and is defined as

$$D(i) = v(i)/[f(i) + 1]$$

- where v(i) is the number of input and output variables that are passed to and from module i.

System complexity is defined as the sum of structural and data complexity, specified as

$$C(i) = S(i) + D(i)$$

Class Oriented Metrics

The CK(The metrics proposed by Chidamber and Kemerer now referred to as the CK metrics), Metrics Suite comprises of Six class-based design metrics for OO systems namely:

❑Weighted methods per class (WMC):-

- The normalized complexity of the methods in a class
- Indicates the amount of effort to implement and test a class

❑Depth of the inheritance tree(DIT):-

- The maximum length from the derived class (the node) to the base class (the root)
- Indicates the potential difficulties when attempting to predict the behavior of a class because of the number of inherited methods

Class Oriented Metrics

- **Number of children(NOC) (i.e., subclasses):-**

  As the number of children of a class grows

  - Reuse increases
  - The amount of testing required will increase

- **Coupling between object classes(CBO)**

  - Measures collaboration between classes
  - Coupling should be kept low, since higher coupling decreases the reusability of a class

Class Oriented Metrics

❑Response for a Class (RFC)

- This is the set of methods that can potentially be executed in a class in response to a public method call from outside the class
- As the response value increases, the effort required for testing also increases as does the overall design complexity of the class

❑Lack of Cohesion in Methods (LCOM)

- This measures the number of methods that access one or more of the same instance variables (i.e., attributes)of a class
- If no methods access the same attribute, then the measure is zero
- As the measure increases, methods become more coupled to one another via attributes, increasing the complexity of class design

Metrics for Maintenance

$$\text{Metric} = [M_T - (F_a + F_c + F_d)]/M_T$$

Where

M_T = number of modules in the current release

F_a = number of modules in the current release that have been added

F_c = number of modules in the current release that have been changed

F_d = number of modules from the preceding release that were deleted in the current release

WebApp Project Metrics

- Number of static Web pages (the end-user has no control over the content displayed on the page)
- Number of dynamic Web pages (end-user actions result in customized content displayed on the page)
- Number of internal page links (internal page links are pointers that provide a hyperlink to some other Web page within the WebApp)
- Number of persistent data objects
- Number of external systems interfaced
- Number of static content objects
- Number of dynamic content objects
- Number of executable functions

Defect Removal Efficiency

$$DRE = E / (E + D)$$

where:

E is the number of errors found before delivery of the software to the end-user

D is the number of defects found after delivery.

Module 4: Software Project Management (13 hrs) – Application level

Project Management Concepts, Project Planning, Overview of metrics, Estimation for Software projects, Project Scheduling, Risk Management, Maintenance and Reengineering, Software Process Improvement (SPI): CMM Levels.

Project Scheduling

Department of Computer Science and Engineering
School of Engineering, Presidency University

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

What is Project Scheduling?

Project scheduling is a mechanism to communicate what tasks need to get done and which organizational resources will be allocated to complete those tasks in what timeframe.

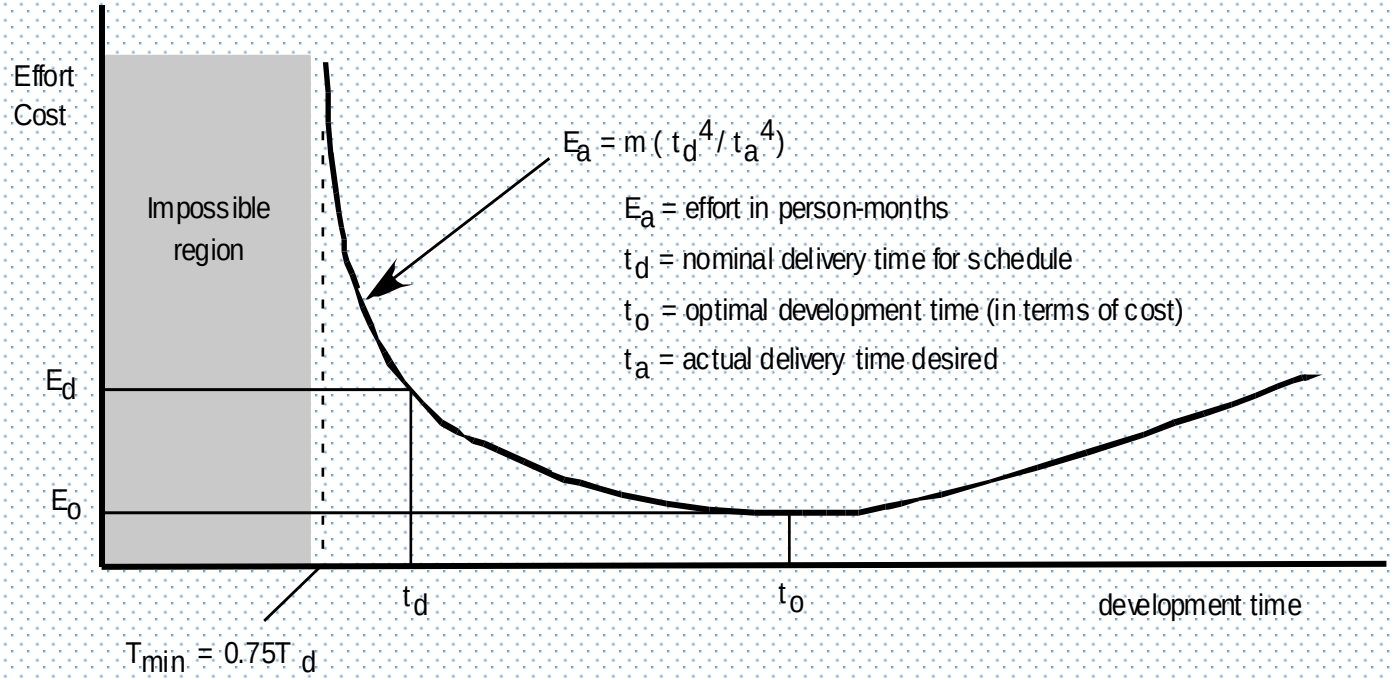
Three important parameters to be considered for Scheduling:

1. Tasks to be completed
2. Timeframe for completion of the tasks
3. Resources who will work on the tasks

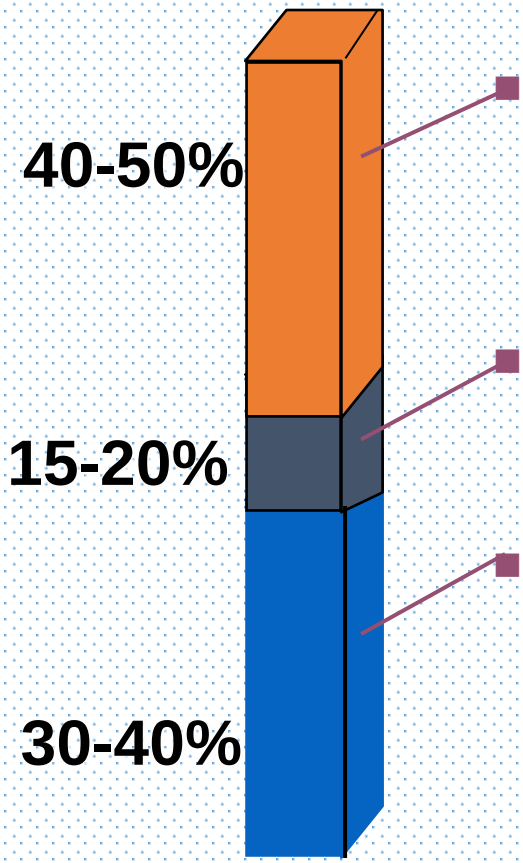
Scheduling Principles

- Compartmentalization - define distinct tasks
- Interdependency - indicate task interrelationship
- Effort validation - be sure resources are available
- Defined Responsibilities - people must be assigned
- Defined Outcomes - each task must have an output
- Defined Milestones - review for quality

Effort and Delivery Time



Typical Effort Allocation across different Project Activities



- "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

Gantt Chart

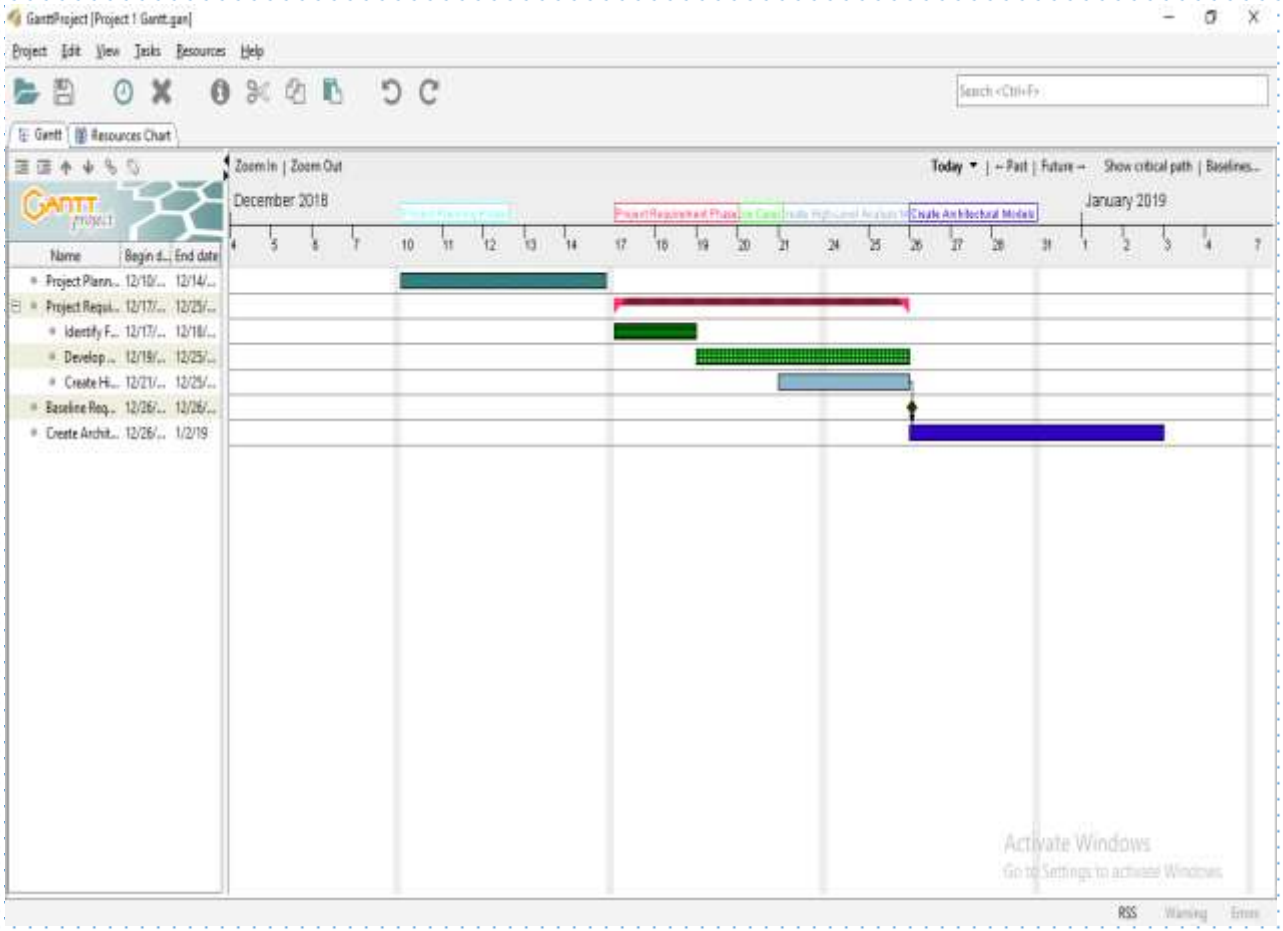
- A **Gantt chart** is a horizontal bar chart developed as a production control tool in 1917 by Henry L. **Gantt**, an American engineer and social scientist. Frequently used in project management, a **Gantt chart** provides a graphical illustration of a schedule that helps to plan, coordinate, and track specific tasks in a project.

Creating a Gantt Chart to depict Project Schedule

Steps:

1) **Create Project Tasks** - Assign start date, duration, priority
2) **Create Project Milestones** - A project milestone is a task of zero duration that shows an important achievement in a project.
3) **Organize tasks** in a work breakdown structure.
4) **Create and Assign Resources** to Tasks
5) **Draw Dependency Constraints** on Tasks

Demo of Gantt Chart using "Gantt Project" software



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 Gantt chart
- Meet informally with project members 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."

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.

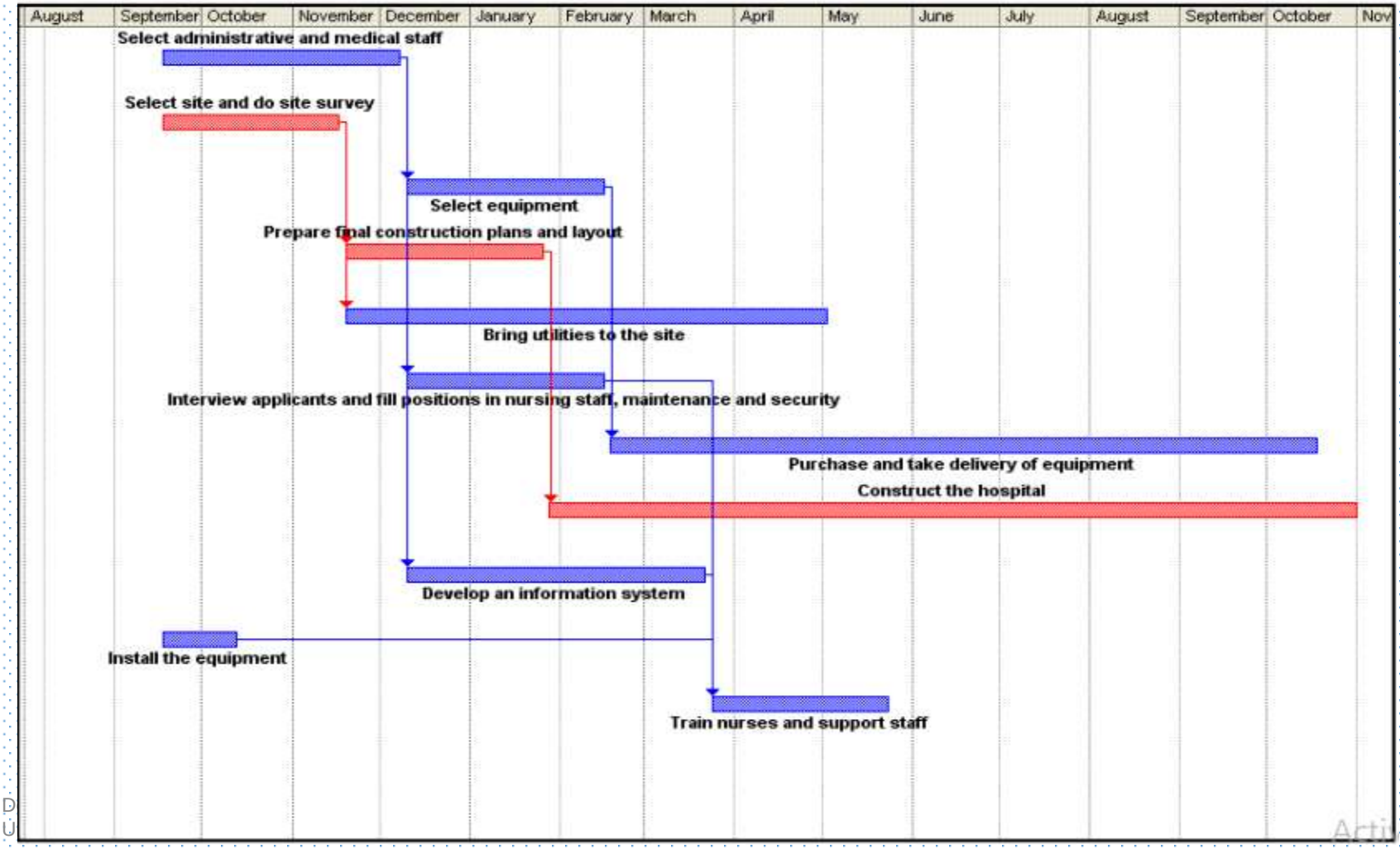
Computing Earned Value-III

- 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

Case Study to create Gantt Chart – Hospital Construction Project

Activity	Description	Predecessors	Duration (Weeks)
A	Select administrative and medical staff.	-	12
B	Select site and do site survey.	-	9
C	Select equipment.	A	10
D	Prepare final construction plans and layout.	B	10
E	Bring utilities to the site.	B	24
F	Interview applicants and fill positions in nursing support staff, maintenance, and security.	A	10
G	Purchase and take delivery of equipment.	C	35
H	Construct the Hospital	D	40
I	Develop an information system	A	15
J	Install the Equipment	E,G,H	4
K	Train nurses and support staff	F,I,J	9

Gantt Chart for the Case



Module 4: Software Project Management (13 hrs) – Application level

Project Management Concepts, Project Planning, Overview of metrics, Estimation for Software projects, Project Scheduling, Risk Management, Maintenance and Reengineering, Software Process Improvement (SPI): CMM Levels.

Risk Management

Department of Computer Science and Engineering
School of Engineering, Presidency University

What is Risk?

Risk indicates "uncertainty" in Projects

Risk increases the probability of suffering loss

An example would be that a team is working on a project and the main developer leaves the company at *crucial stage* of the project. The alternate resource replacing him will take time to adjust.

Whether the team will be able to complete the project on time? That is the risk of schedule.

Types of Risk Management

1. Reactive Risk Management
2. Proactive Risk Management

Reactive Risk Management

- Project team reacts to risks when they occur
- Mitigation - plan for additional resources in anticipation of fire fighting
- Fix on failure - resources are found and applied when the risk strikes
- Crisis management - failure does not respond to applied resources and project is in jeopardy

Types of Risk Management

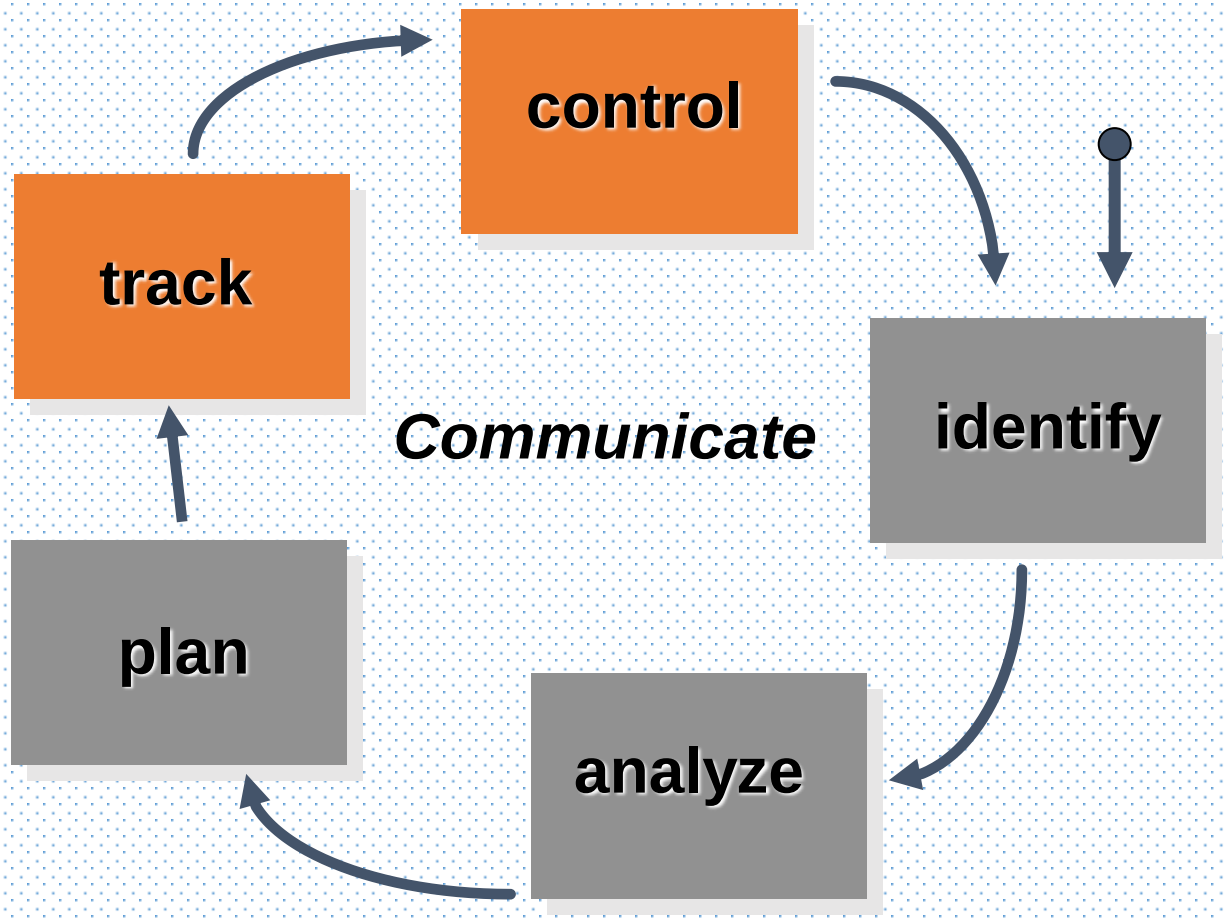
Proactive Risk Management

- Formal risk analysis is performed
- Organization corrects the root causes of risk
  - Examining risk sources that lie beyond the bounds of the software
  - Developing the skill to manage change

Seven Principles of Risk Management

1. **Maintain a global perspective** - view software risks within the context of system and the business problem
2. **Take a forward-looking view** - think about the risks that may arise in the future; establish contingency plans
3. **Encourage open communication** - if someone states a potential risk, don't discount it.
4. **Integrate** - a consideration of risk must be integrated into the software process
5. **Emphasize a continuous process** - the team must be vigilant throughout the software process, modifying identified risks as more information is known and adding new ones as better insight is achieved.
6. **Develop a shared product vision** - if all stakeholders share the same vision of the software, it likely that better risk identification and assessment will occur.
7. **Encourage teamwork** - the talents, skills and knowledge of all stakeholder should be pooled

Risk Management Paradigm



- Identify: Search for the risks before they create a major problem
- Analyze: understand the nature , kind of risk and gather information about the risk.
- Plan: convert them into actions and implement them.
- Track: we need to monitor the necessary actions.
- Control: Correct the deviation and make any necessary amendments.
- Communicate: Discuss about the emerging risks and the current risks and the plans to be undertaken.

Risk Components

1. *Performance risk* - the degree of uncertainty that the product will meet its requirements and be fit for its intended use.
2. *Cost risk* - the degree of uncertainty that the project budget will be maintained.
3. *Support risk* - the degree of uncertainty that the resultant software will be easy to correct, adapt, and enhance.
4. *Schedule risk* - the degree of uncertainty that the project schedule will be maintained and that the product will be delivered on time.

Risk Projection

- *Risk projection*, also called *risk estimation*, attempts to rate each risk in two ways
  - the likelihood or probability that the risk is real
  - the consequences of the problems associated with the risk, should it occur.
- The are four risk projection steps:
  - establish a scale that reflects the perceived likelihood of a risk
  - delineate the consequences of the risk
  - estimate the impact of the risk on the project and the product,
  - note the overall accuracy of the risk projection so that there will be no misunderstandings.

Building a Risk Table

Risk	Probability	Impact	RMMM
			Risk Mitigation Monitoring & Management

Building the Risk Table

- Estimate the probability of occurrence
- Estimate the impact on the project on a scale of 1 to 5, where
  - 1 = low impact on project success
  - 5 = catastrophic impact on project success
- sort the table by probability and impact

Risk Exposure (Impact)

- The overall *risk exposure,* RE, is determined using the following relationship [Hal98]:
- **RE = *P* x *C***
- where
- *P* is the probability of occurrence for a risk, and
- *C* is the cost to the project should the risk 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.

Case Study

Lupus is a chronic autoimmune disease that can damage any part of the body (skin, joints, and/or organs). Nearly 0.32% of Indian population suffers from this disease.

Imagine you are the Project Manager for developing a healthcare application that collects, analyzes and classifies blood samples into Lupus and Non-Lupus categories.

Identify the risk factors for this application?

Case Study

Some risk factors for the healthcare application includes:

- **Technology risks** (Machine Learning algorithms not mature enough for classification into Lupus and non-lupus categories yet)
- **Compliance risks** (Getting Patient consent for using his/her data in the project)
- **Competition risks** (Competitor may come out with similar project and release into market first)
- **Business risks** (Identify which hospitals will be prospective buyers or users of this application)
- **Project Team related risk** (Employee leaving project; must maintain confidentiality regarding development secrets)
- **Cost risks** (Overshooting budget if enough samples are not available)

Module 4: Software Project Management (13 hrs) – Application level

Project Management Concepts, Project Planning, Overview of metrics, Estimation for Software projects, Project Scheduling, Risk Management, Maintenance and Reengineering, Software Process Improvement (SPI): CMM Levels.

R&D SDM 1

Software Process Improvement Capability Maturity Models

2010

Theo Schouten

Content

- Process, product, people, quality
- What are maturity models?
- CMMI models
- Operational use of the CMMI
- People-CMM

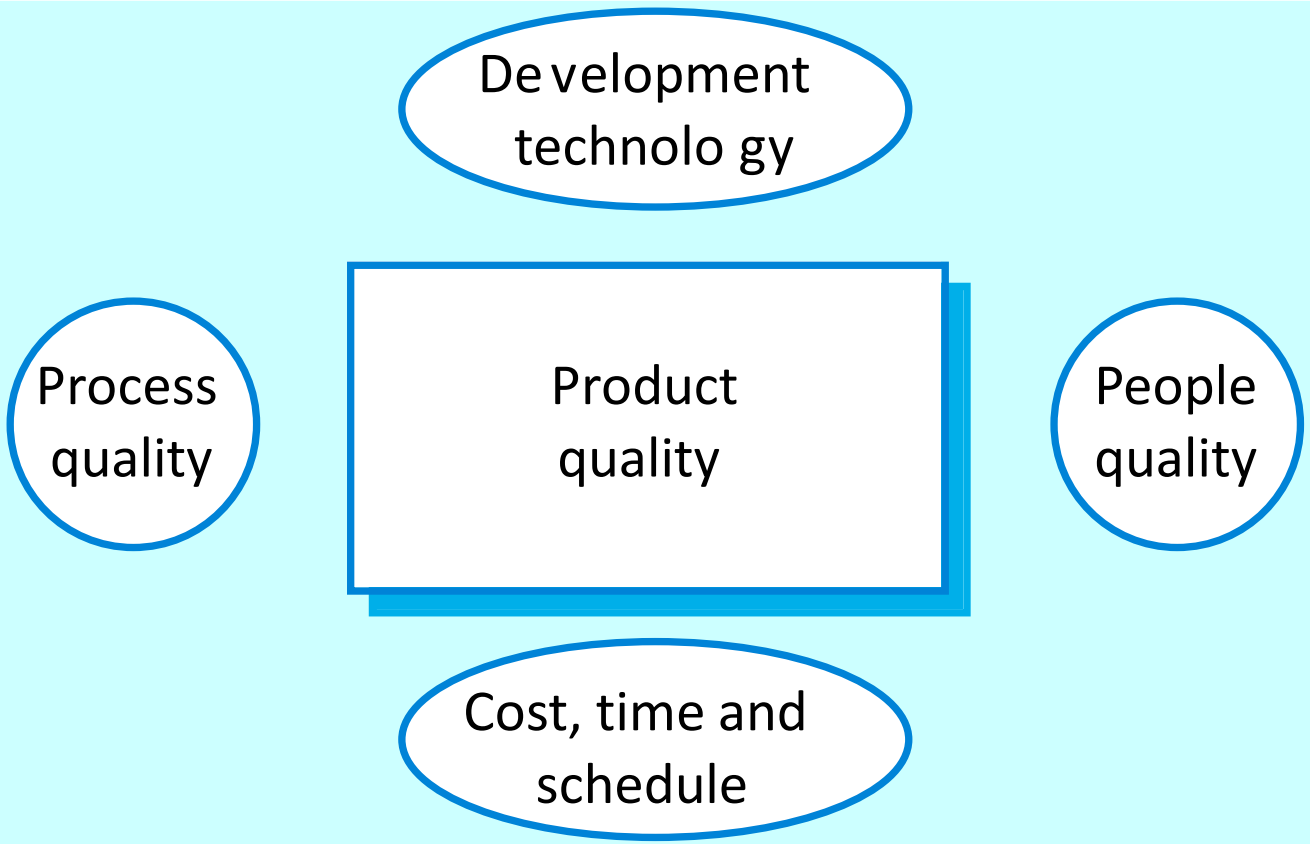
Book chapter 2: Process: A Generic View

7: chapter 30 new

Process and product quality

- A good process is usually required to produce a good product.
- Improvement of the process give benefits because the quality of the product depends on its development process.
- For industrial production, process is the principal quality determinant.
- For design-based activities, the capabilities of the designers are also an important factor

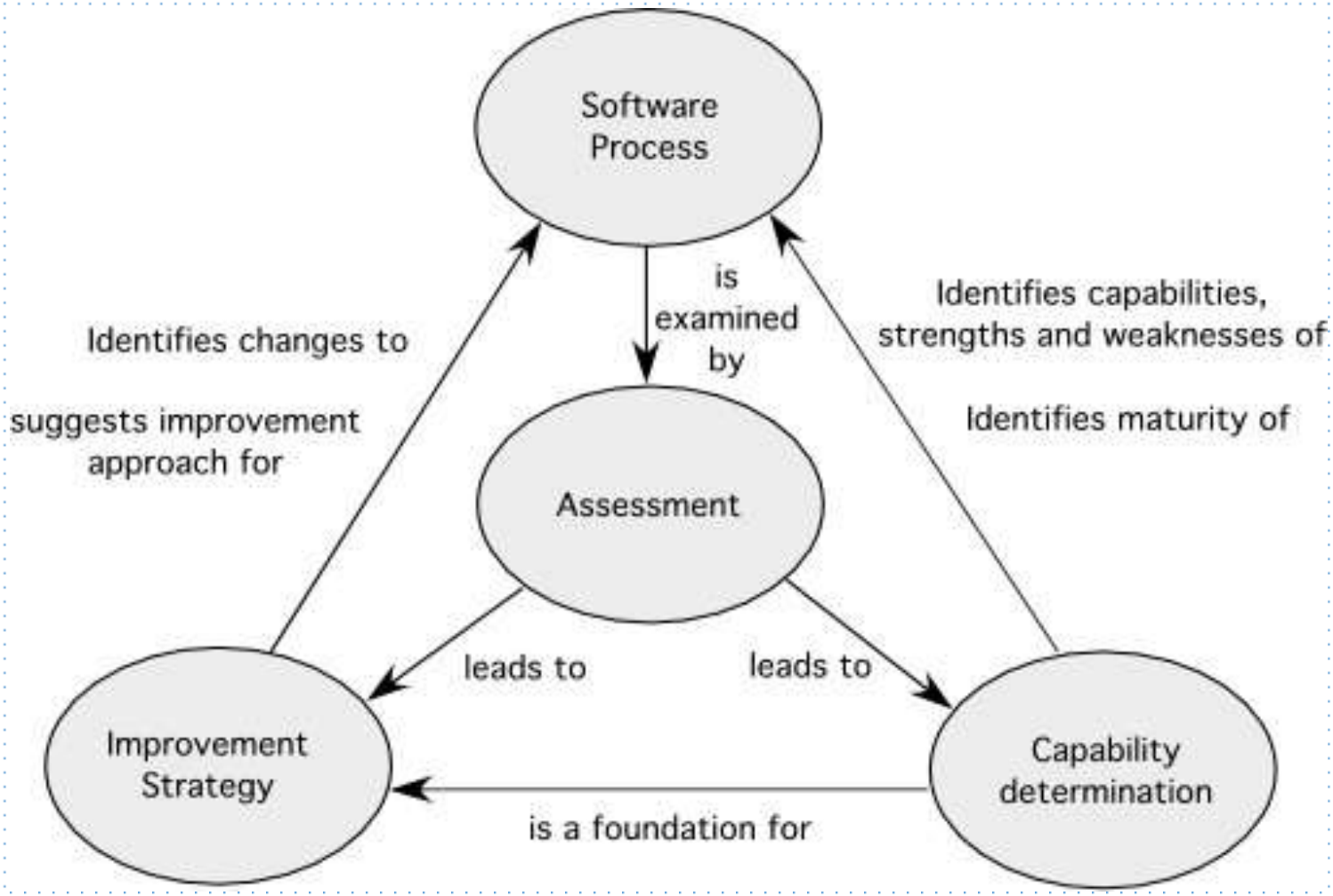
Principal product quality factors



Software Process Improvement stages

- Process measurement
  - Attributes of the current process are measured. These are a baseline for assessing improvements
- Process analysis
  - The current process is assessed and bottlenecks and weaknesses are identified
- Process change
  - Changes to the process that have been identified during the analysis are introduced

Elements of an SPI framework



What is a maturity model?

'A maturity model is a model with which organizations can **judge** their (software engineering, hrm, service, etc.) process (including comparing it to other organizations) and based on this judgment can **improve** their process.'

– started in 1986 by the Software Engineering Institute (SEI) (Carnegie Mellon University) and the Mitre Corporation

–SEI started with a Process Maturity Framework and a maturity questionnaire

–the software Framework developed into the Capability Maturity Model (CMM) for Software (1991)

–Revised maturity framework (CMMI) introduced in 2001

Process Maturity Framework

- Goal is the improvement of the software engineering process
  - Success should not be based on incidental individual achievements
  - Success should be based on repeatable and proven successful work methods

Immature Software Organization

- No objective basis for judgment of product quality
- No objective basis for improvement of product or process quality
- Reviews and tests are often skipped when projects run behind schedule
- Ad hoc management

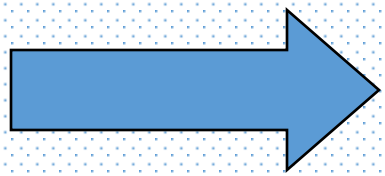
Mature Software Organization

- Organization wide knowledge to manage software development, employment and improvement
- Processes are ‘fit-for-use’
- Processes are being adapted to the situation
- Tasks and responsibilities are clear for the project and anyone in the organization

Software Process

Software Process:

‘the whole of activities, methods, practices, communication and changes that people use in order to develop and maintain software and associated products (e.g. plans, design documents, code, test cases and user manuals)’



CMM gives an organization a way to get and improve control over the software process and provides it with a route to achieve excellence in software engineering

Fundamental concepts for Process Maturity

- Software Process Capability
  - –How good it can predict the expected outcome of a next software project
- Software Process Performance
  - –Actual results of a software project
- Software Process Maturity
  - –The level in which a software process is explicitly defined, managed, measured and controlled in order to achieve results
- Software Process Institutionalization
  - –The level in which the software process is institutionalized with respect to methods, standards and organizational structure

Levels of software maturity

Maturity level:

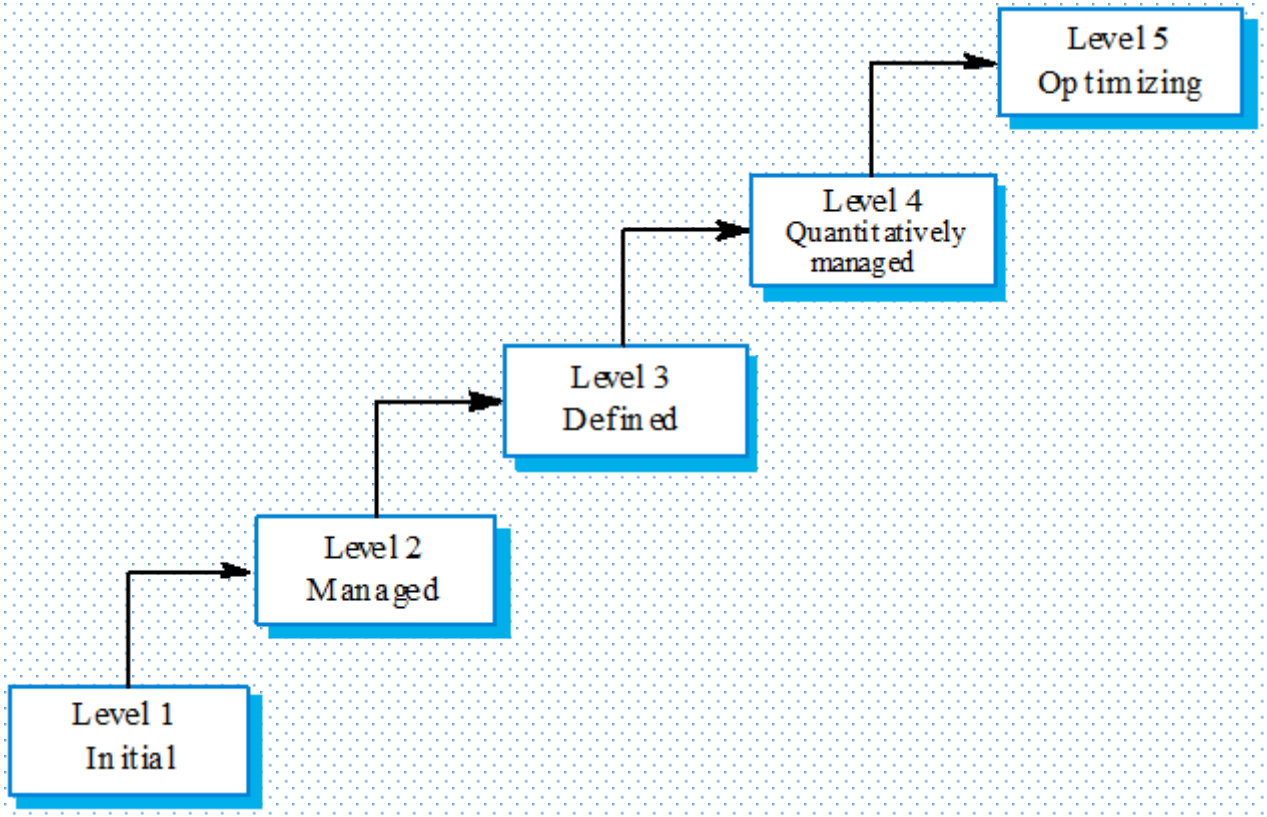
–A well defined level on the way to achieve an adult, a mature software process

–A foundation for realizing continuous improvements

–Every level contains a group of process goals that, when stable, form an important part of the software process

–Every level leads to the improvement of the process capability of the organization

The staged CMMI model



Initial (1)

- The software process can be described as ad-hoc, or even chaotic
- There are practically no processes defined
- Success depends on individual input and achievements
- The software process is not predictable regarding results
- Schedules, budget, functionality and product quality is not predictable
- Works disastrous in crises situations
- Can be successful in highly innovative environment (e.g. start of the web-design world)

Managed (2)

- The basic project management procedures are used
- Costs, schedules en functionality are ‘tracked’
- Planning and managing of new projects are based on experience with comparable projects
- Needed process discipline is enforced such that earlier success can be repeated with building an comparable application
- Software requirements and work products are ‘baselined’
- Disciplined environment in which planning and tracking are stable and thus previous successes can be repeated

Defined (3)

- Processes for management and software engineering are documented, standardized and integrated in a standard software development process
- All projects use an approved, adapted version of the standard software process for the development and maintenance of software
- Processes are used to let software managers and engineers be more effective
- There is a group responsible for the software process
- There is training in the software process
- The software process is stable and well defined and is able to operate more effectively

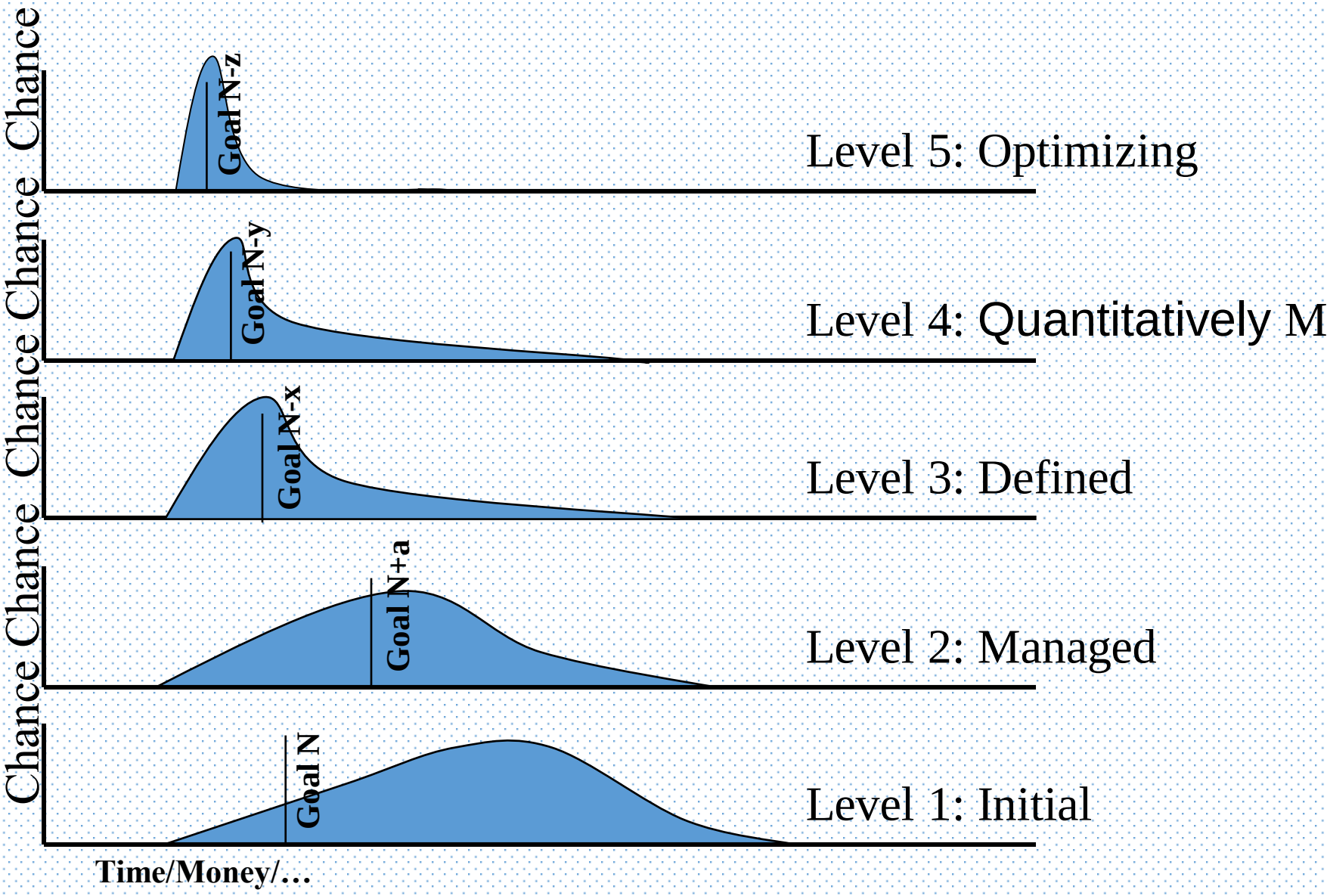
Quantitatively managed (4)

- Detailed metrics of the software processes and quality of products are gathered
- Quantitative goals are set for the software process and the product quality
- Use is made of a software process database in which the metrics are gathered and analyzed
- Projects have a control over the software process and product quality such that the can work in defined limits
- Risks of development in new technical environments are recognized and managed
- Software process is predictable and trends can be predicted

Optimizing (5)

- Continuously software process improvements are realized by quantitatively feedback of the process and by trying out of innovative ideas and technologies
- The whole organization is focused on continuous improvement
- Data regarding performance of the processes are used for cost-benefit analyses
- Innovations that make use of the best software engineering practices are identified and spread over the whole organization
- Software project teams analyze errors in order to find out how to improve
- ‘Lessons learned’ are shared with other projects (team rooms, Communities of Practice)

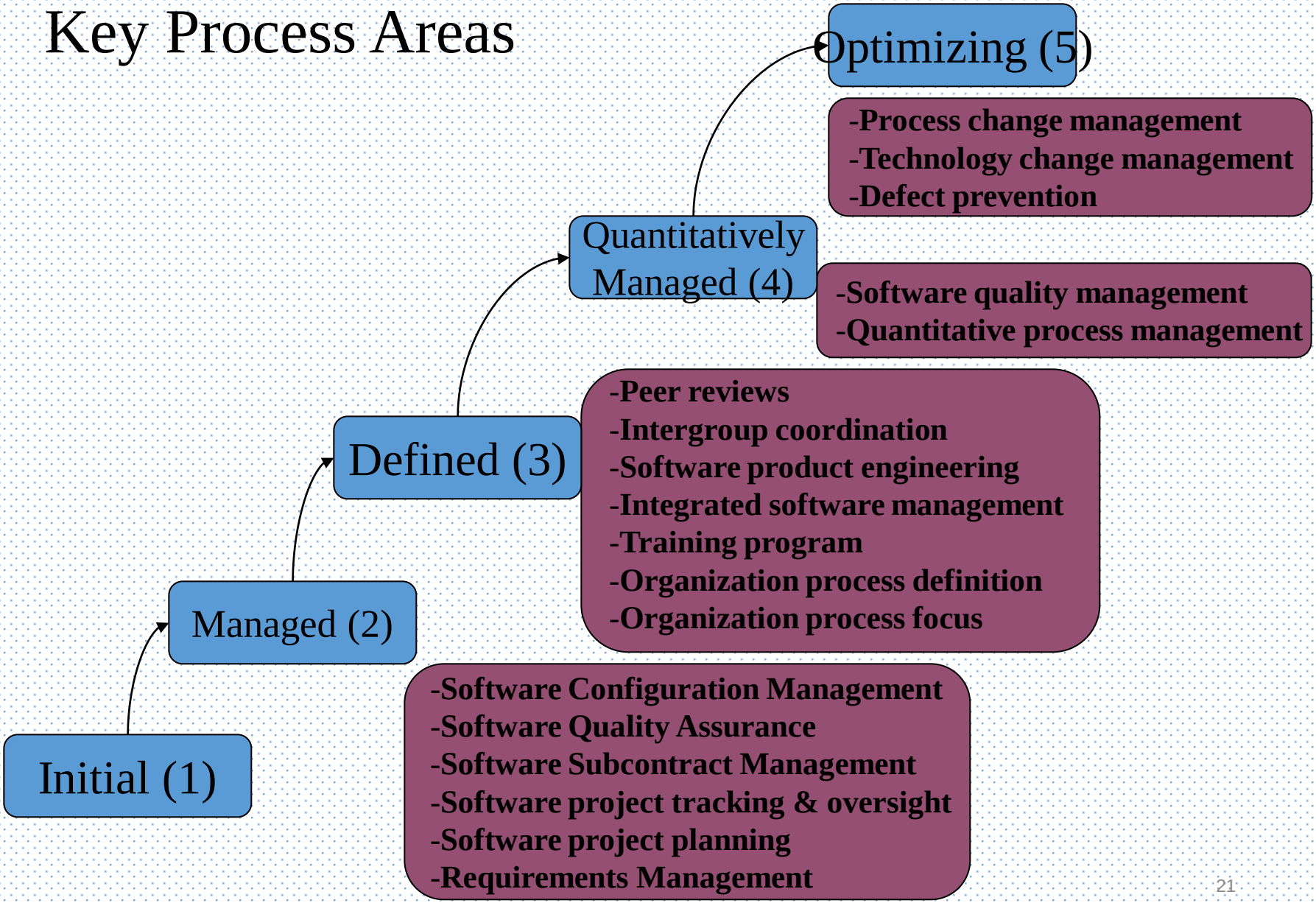
Maturity level and changing predictability



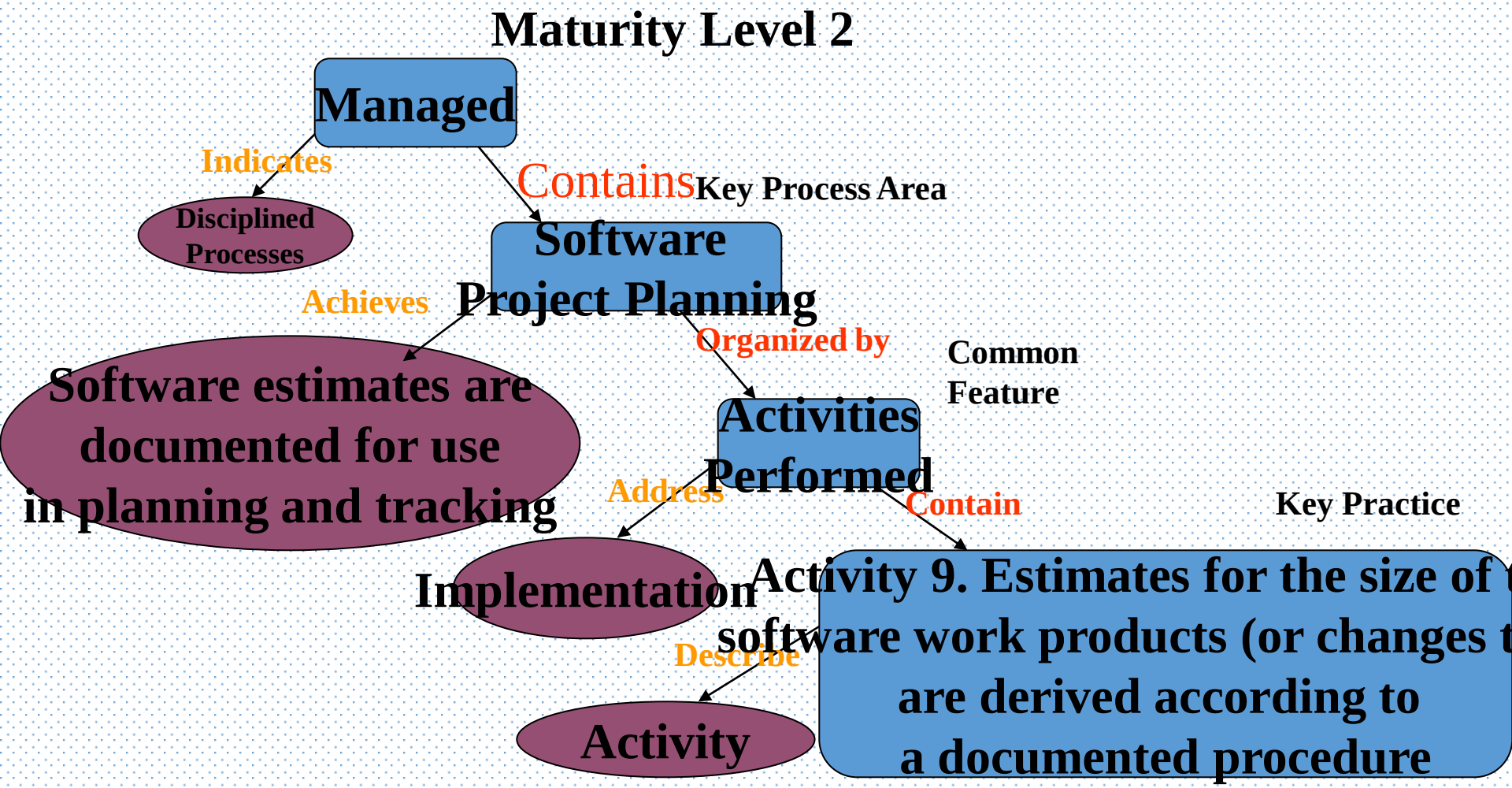
Operational use of CMM

How do you determine in practice the maturity of an organization?

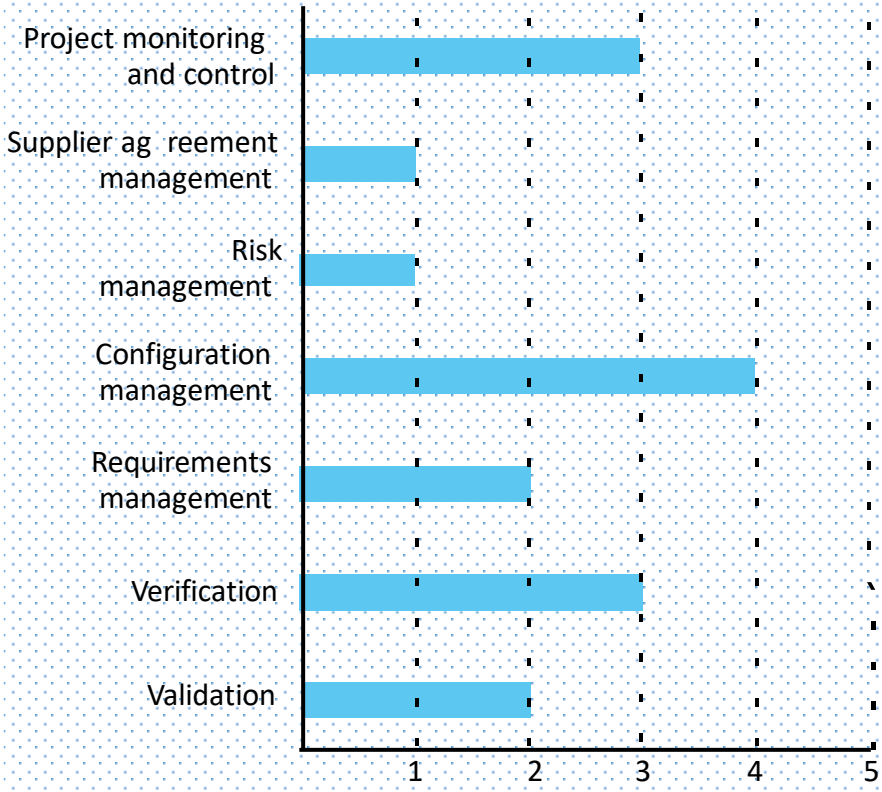
Key Process Areas



Example


Maturity Level 2
Managed
Indicates
Disciplined Processes
Contains
Key Process Area
Software Project Planning
Achieves
Software estimates are documented for use in planning and tracking
Organized by
Common Feature
Activities Performed
Address
Implementation
Contain
Key Practice
Activity 9. Estimates for the size of t
software work products (or changes t
are derived according to
a documented procedure
Describe
Activity

Continuous CMM model



A process capability profile

Remarks

- Maturity Models are helpful to indicate the maturity of a software organization
- The CMM(I) model is the most used
- Organizations ‘benchmark’ themselves to position them relative to others based on the CMM-level
- CMM-level should give an indication of the quality level of a software organization
- Specially “new” countries (India, China) qualify themselves strongly in this area
- It is not a “sacred cow” and it should be used prudently

Problems with CMM

- CMM is actually a management framework, with many details left out (a goal, not a method)
  - Example: “You must have peer reviews.” But how should the reviews be run?
- Being used just as stamp of approval
  - Just tell us what to do to get Level 2, so we can get back to work
  - Let’s work together to improve our software processes
- Doesn’t say anything about software!
  - CMM is a model for *managing* software projects
- Doesn’t help in a crisis
- Only for repetitive tasks

People- Capability Maturity Model (P-CMM)

- Improvement of the capabilities of the software organization by improvement of the skills of the individuals
- Assure that the ability in software development is an attribute of the organization and not of a number of individuals
- Keeping people with critical knowledge and skills in the organization
- Assure that the goals and direction of the individuals is the same as that of the organization

Provides

- A framework that focuses on improving management and development of human assets of an organization.
- Provides an improvement path
- Integrate workforce development with process improvement
- Characterize the maturity of their workforce practices

Trends Affecting the Workforce

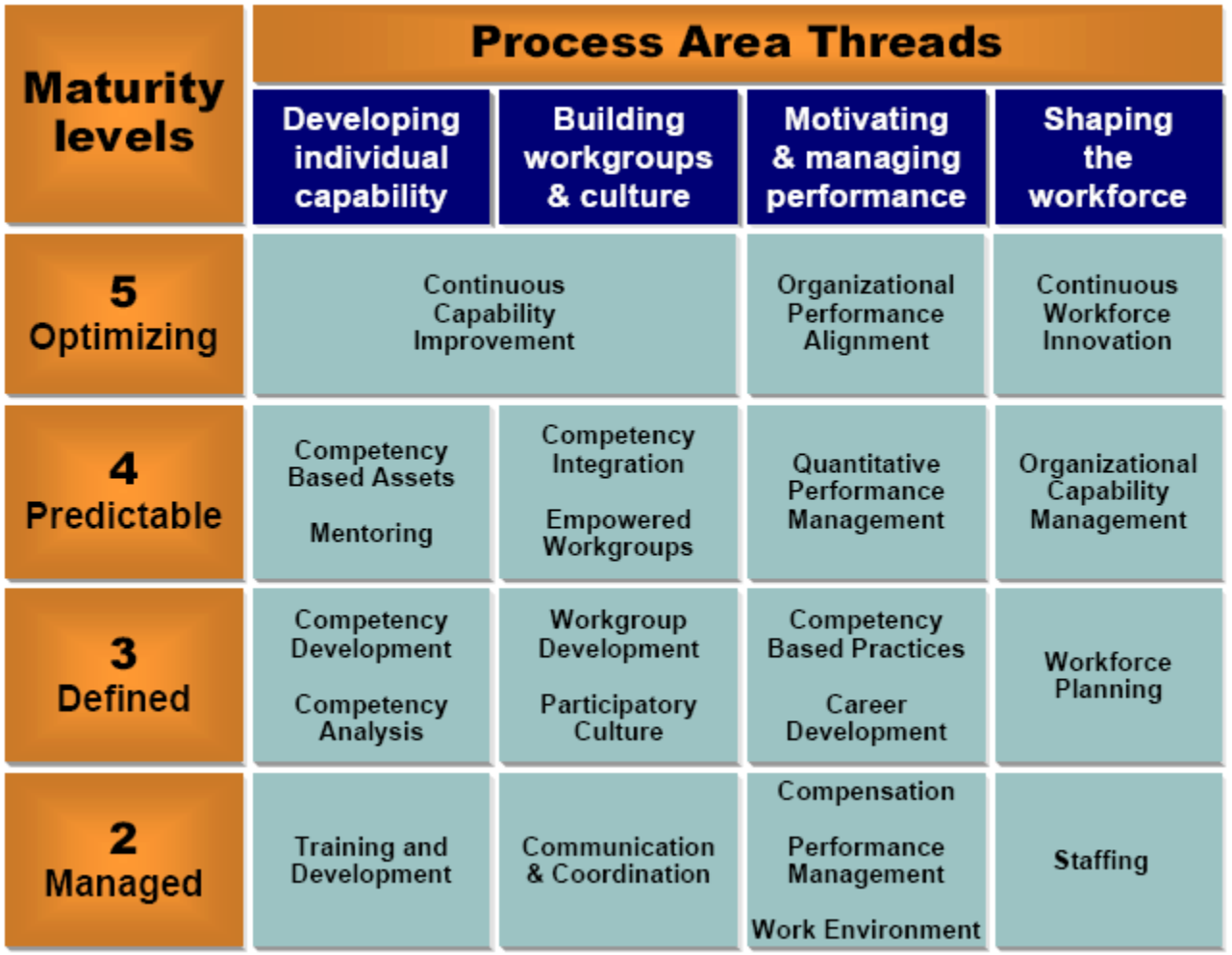
- *Pfeffer (1994)*
- Doers differ from thinkers -> Doers must be thinkers
- Assets are things -> Assets are people
- Labor is an expense -> People are an investment
- Lifetime employment -> Lifetime employability
- Top down control -> Decentralized decisions
- Localized work -> Networked problems solved
- Measure for results -> Measure for improvements

P-CMM Architecture

P-CMM levels and process areas

Level	Focus	Process Areas
Optimized	*Continuous improvement*	Continuous workforce innovation Organizational performance alignment Continuous capability improvement
Predictable	*Quantifies and manages knowledge, skills and abilities*	Mentoring Organizational capability management Quantitative performance management Competency-based assets Empowered workgroups Competency integration
Defined	*Identifies and develops knowledge, skills and abilities*	Participatory culture Workgroup development Competency-based practices Career development Competency development Workforce planning Competency analysis
Managed	*Repeatable, basic people management practices*	Compensation Training and development Performance management Work environment Communication and coordination Staffing
Initial	*Inconsistent practices*	

Process Area & Maturity Levels



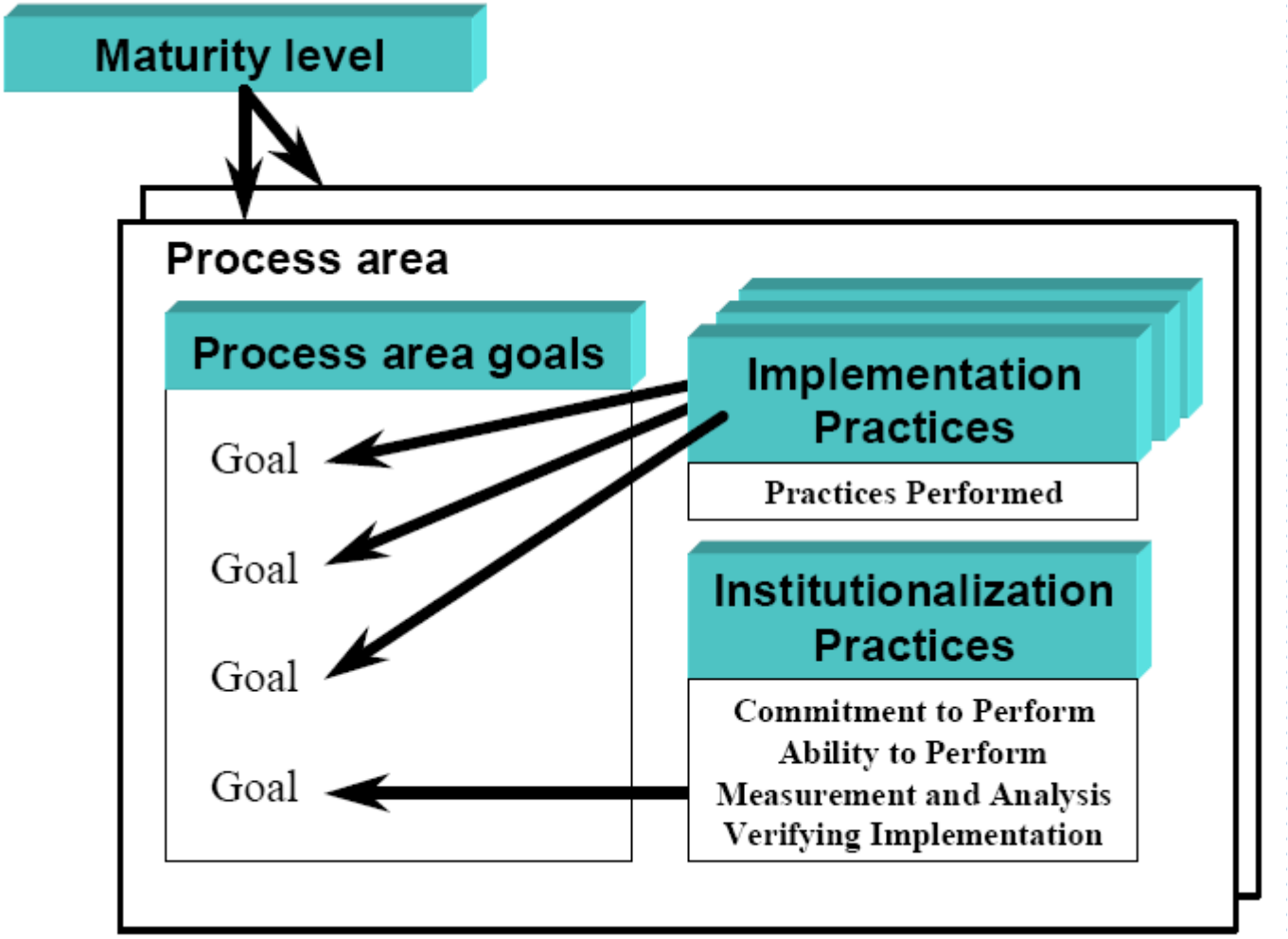
Maturity levels	Process Area Threads			
	Developing individual capability	Building workgroups & culture	Motivating & managing performance	Shaping the workforce
5 Optimizing	Continuous Capability Improvement		Organizational Performance Alignment	Continuous Workforce Innovation
4 Predictable	Competency Based Assets Mentoring	Competency Integration Empowered Workgroups	Quantitative Performance Management	Organizational Capability Management
3 Defined	Competency Development Competency Analysis	Workgroup Development Participatory Culture	Competency Based Practices Career Development	Workforce Planning
2 Managed	Training and Development	Communication & Coordination	Compensation Performance Management Work Environment	Staffing

Process Goals & Process Practice

- Process area goal is an organizational state to be achieved by implementing the practices of a process area.

- Process practice is a sub process within a process area that contributes to achieving a process area goal.

Implementation & Institutionalization Practices



Other SPI frameworks

- **SPICE**— a international initiative to support the International Standard ISO/IEC 15504 for (Software) Process Assessment [ISO08]
- **Bootstrap**—a SPI framework for small and medium sized organizations that conforms to SPICE [Boo06],
- **PSP and TSP**—individual and team specific SPI frameworks ([Hum97], [Hum00]) that focus on process in-the-small, a more rigorous approach to software development coupled with measurement
- **TickIT**—an auditing method [Tic05] that assesses an organization compliance to ISO Standard

SPI trends

- future SPI frameworks must become significantly more agile
- Rather than an organizational focus (that can take years to complete successfully), contemporary SPI efforts should focus on the project level
- To achieve meaningful results (even at the project level) in a short time frame, complex framework models may give way to simpler models.
- Rather than dozens of key practices and hundreds of supplementary practices, an agile SPI framework should emphasize only a few pivotal practices