Software Engineering

Chapter 3: Project Planning and Risk Management

Mekia Shigute Gaso

mekiashigute.gaso@alatoo.edu.kg

Faculty of Engineering and Computer Science
Department of Computer Science
Ala-Too International University

December 8, 2025

3.1: Software Project Planning

- **A Software Project is the complete methodology of programming advancement from that of requirement gathering all the way to testing and supporting it.**
- It is therefore completed by the execution of procedures in a quite specified period to achieve the intended software product.

Need of Software Project Management

- Today software development involves almost all new streams in the world business.
- Most programming items are customized to accommodate customer's necessities.
- The most significant reason is that the underlying technology changes and advances so generally and rapidly that experience of one element may not be connected to the other one. Thus we need software project management which is used for putting all these together.
- All such business and ecological imperatives bring risk in software development; hence, it is fundamental to manage software projects efficiently.

Software Project Manager

- Software manager is responsible for planning and scheduling project development.
- They manage the work to ensure that it is completed to the required standard.
- They monitor the progress to check that the event is on time and within budget.
- The project planning must incorporate the major issues like size cost estimation scheduling, project monitoring, personnel selection evaluation risk management.
- To plan a successful software project, we must understand:

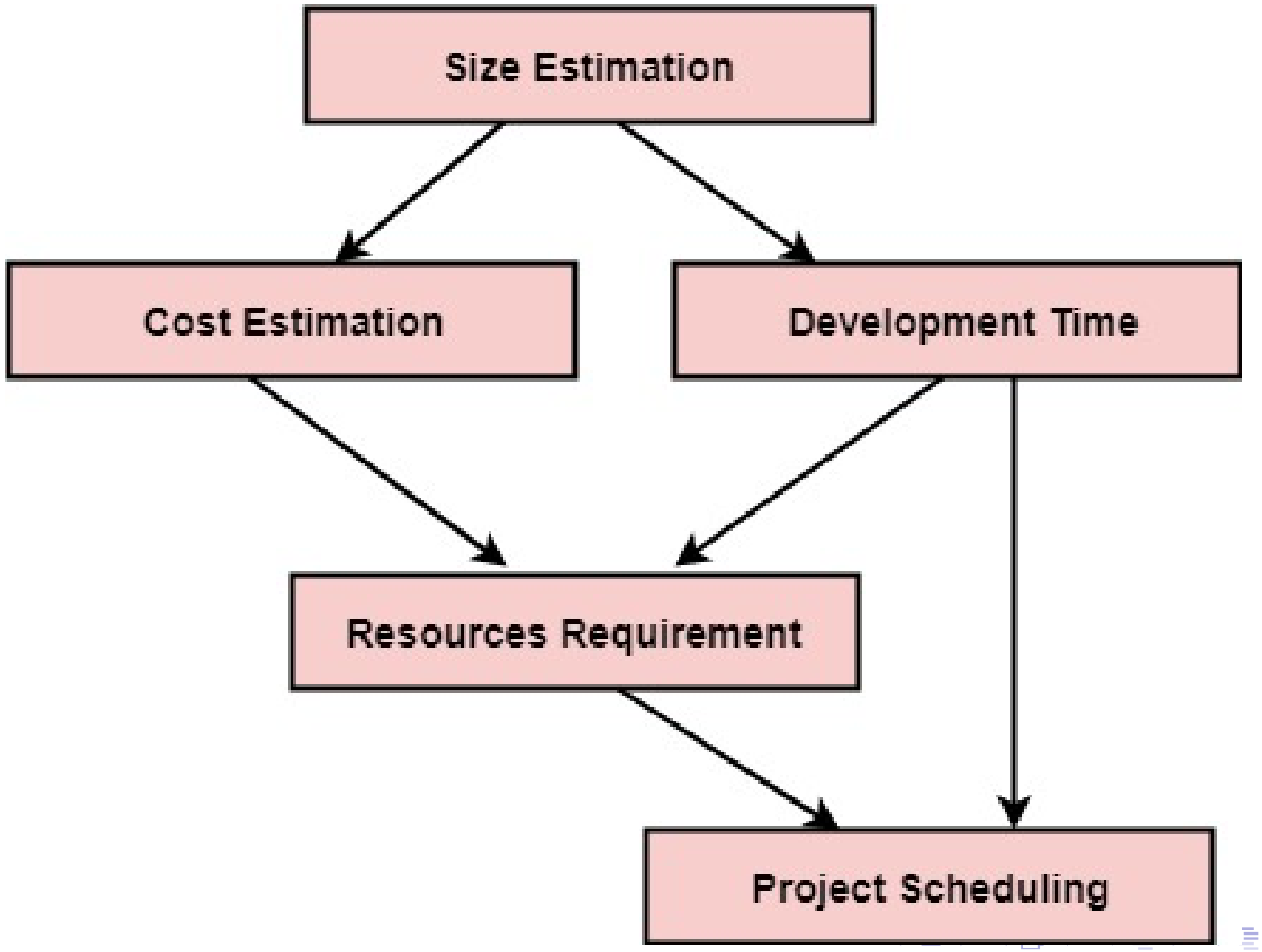
Software Project Manager (cont.)

- **To plan a successful software project, we must understand:**
  - Scope of work to be completed.
  - Risk analysis.
  - The resources which are mandatory.
  - The project to be accomplished.
  - Record of what is being followed.

Software Project Manager (cont.)

Software Project planning starts before technical work start.

The various steps of planning activities are:



Software Project Manager (cont.)

- The size is the crucial parameter for the estimation of other activities.
- Resources requirement are required based on cost and development time.
- Project schedule may prove to be very useful for controlling and monitoring the progress of the project.
- This is dependent on resources & development time.

3.1.1: Software Cost Estimation

- For any new software project, it is necessary to know how much it will cost to develop and how much development time will it take.
- These estimates are needed before development is initiated, but how is this done?
- Several estimation procedures have been developed and are having the following attributes in common.

Software Cost Estimation (cont.)

- Project scope must be established in advanced.
- Software metrics are used as a support from which evaluation is made.
- The project is broken into small PCs which are estimated individually.
- To achieve true cost & schedule estimate, several option arise.
- Delay estimation
- Used symbol decomposition techniques to generate project cost and schedule estimates.
- Acquire one or more automated estimation tools.

Uses of Cost Estimation

- During the planning stage, one needs to choose how many engineers are required for the project and to develop a schedule.
- In monitoring the project's progress, one needs to access whether the project is progressing according to the procedure and takes corrective action, if necessary.

Cost Estimation Models

- Cost estimation models can be static or dynamic one.
- In a static model, a single variable is taken as a key element for calculating cost and time.
- In a dynamic model, all variable are interdependent, and there is no basic variable.

Cost Estimation Models

- When a model makes use of single variables to calculate desired values such as cost, time, efforts, etc. is said to be a single variable model.
- The most common equation is:

$$C = aL^b,$$

where C = Costs, L= size, a and b are constants.

- The Software Engineering Laboratory established a model called SEL model, for estimating its software production.

Cost Estimation Models (cont.)

- This model is an example of the static, single variable model.

$$E = 1.4L^{0.93}$$

$$DOC = 30.4L^{0.90}$$

$$D = 4.6L^{0.26},$$

where E= Efforts (Person Per Month), DOC=Documentation (Number of Pages), D = Duration (D, in months) and L = Number of Lines per code.

Cost Estimation Models (cont.)

Static, Multivariable Models:

- These models are based on method (1), they depend on several variables describing various aspects of the software development environment.
- In some model, several variables are needed to describe the software development process, and selected equation combined these variables to give the estimate of time cost.
- These models are called multivariable models.

Walston-Felix Model

WALSTON and FELIX develop the models at IBM provide the following equation gives a relationship between lines of source code and effort:

$$E = 5.2L^{0.91}$$

In the same manner duration of development is given by

$$D = 4.1L^{0.36}$$

Walston-Felix Model (cont.)

The productivity index uses 29 variables which are found to be highly correlated productivity as follows:

$$I = \sum_{i=1}^{29} W_i X_i,$$

where W_i is the weight factor for the i^{th} variable and $X_i = -1, 0, +1$ the estimator gives X_i one of the values -1, 0 or $+1$ depending on the variable decreases, has no effect or increases the productivity.

Example: Walston-Felix vs SEL Models

- **Example: Compare the Walston-Felix Model with the SEL model on a software development expected to involve 8 person-years of effort.**
  - Calculate the number of lines of source code that can be produced.
  - Calculate the duration of the development.
  - Calculate the productivity in LOC/PY.
  - Calculate the average number of persons required per month in the project.

Solution: Walston-Felix vs SEL Models

- **The amount of manpower involved is:** 8 PY (person-year) = 96 (8×12) persons-months.
- **[I] Number of lines of source code can be obtained by reversing equation to give:**

$$L = (\frac{E}{a})^{1/b}$$

  - For SEL model:

$$L_{SEL} = (\frac{E}{a})^{1/b} = (\frac{96}{1.4})^{1/0.93} = 94.264 \text{ LOC}$$

  - For Walston-Felix (W-F) model:

$$L_{W-F} = (\frac{E}{a})^{1/b} = (\frac{96}{5.2})^{1/0.91} = 24.632 \text{ LOC}$$

Solution: Walston-Felix vs SEL Models (cont.)

- **[II] Duration in months can be calculated by means of equation**

$$D_{SEL} = 4.6 L_{SEL}^{0.26} \quad \text{and} \quad \mathrm{D_{W-F}} = 4.1 \mathrm{L_{W-F}^{0.36}},$$

for SEL and W-F model, respectively

- Thus for SEL model we have:

$$D_{SEL} = 4.6 L^{0.26} = (4.6) \times L_{SEL}^{0.26} = (4.6) \times (94.264)^{0.26} = 15 \text{ Months}$$

- And for W-F model we have:

$$D_{W-F} = 4.1 L^{0.36} = (4.1) \times L_{W-F}^{0.36} = (4.1) \times (24.632)^{0.36} = 13 \text{ Months}$$

Solution: Walston-Felix vs SEL Models (cont.)

- **[III] Productivity(P) is the lines of code produced per persons/month (year):**

$$D_{SEL} = \frac{L_{SEL}}{\text{person per month}} = \frac{94.264}{8} = 11.783 \frac{\text{LOC}}{\text{Person}} - Year$$

$$D_{W-F} = \frac{L_{W-F}}{\text{person per month}} = \frac{24.632}{8} = 3.079 \frac{\text{LOC}}{\text{Person}} - Year$$

- **[IV] The average number of person (M) required per month in the project:**

$$M_{SEL} = \frac{\text{Number of person per month}}{\text{Duration of months}} = \frac{96}{15 \text{ Months}} = 6.4 \text{ Person}$$

$$M_{W-F} = \frac{\text{Number of person per month}}{\text{Duration of months}} = \frac{96}{13 \text{ Months}} = 7.4 \text{ Person}$$

3.1.2: COCOMO Model

- Boehm proposed COCOMO (Constructive Cost Estimation Model) in 1981.
- COCOMO is one of the most generally used software estimation models in the world.
- COCOMO predicts the efforts and schedule of a software product based on the size of the software.
- The necessary steps in this model are:
  1. Get an initial estimate of the development effort from evaluation of thousands of delivered lines of source code (KDLOC).
  2. Determine a set of 15 multiplying factors from various attributes of the project.
  3. Calculate the effort estimate by multiplying the initial estimate with all the multiplying factors i.e., multiply the values in step1 and step2.

COCOMO Model (cont.)

- The initial estimate (also called nominal estimate) is determined by an equation of the form used in the static single variable models, using KDLOC as the measure of the size.
- To determine the initial effort E_i in person-months the equation used is of the type which is shown below

$$\boxed{E_i = a * (KDLOC)^b.}$$

- The value of the constant a and b varies depending on the project type used.
- In COCOMO, projects are categorized into three types: Organic, Semidetached and Embedded.
- Let us see each of these one by one:

COCOMO Model Parameters

Software Projects	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

COCOMO Projects [1. Organic]

- A development project can be treated of the organic type if:
  - the project deals with developing a well-understood application program,
  - the size of the development team is reasonably small, and
  - the team members are experienced in developing similar methods of projects.
- Examples of Organic type of projects are
  - simple business systems,
  - simple inventory management systems, and
  - data processing systems.

COCOMO Projects [2. Semidetached]

- A development project can be treated with semidetached type if:
  - the development consists of a mixture of experienced and inexperienced staff.
  - team members possibly may have adequate experience in related systems but may be unfamiliar with some aspects of the order being developed.
- Example of Semidetached system includes developing
  - new operating system (OS),
  - Database Management System (DBMS), and
  - complex inventory management system.

COCOMO Projects [3. Embedded:]

- A development project is treated to be of an embedded type, if
  - the software being developed is strongly coupled to complex hardware, or
  - the stringent (strict) regulations on the operational method exist.
- Example of Embedded types of projects are
  - Automated Teller Machine (ATM),
  - Air traffic control.

COCOMO Projects (cont.)

- For three product categories, Bohem provides a different set of expression to predict effort (in a unit of person month)and development time from the size of estimation in KLOC(Kilo Line of code) efforts estimation takes into account the productivity loss due to holidays, weekly off, coffee breaks, etc.
- According to Boehm, software cost estimation should be done through three stages: Basic Model, Intermediate Model and Detailed Model.

1. Basic COCOMO Model:

- The basic COCOMO model provide an accurate size of the project parameters.
- The following expressions give the basic COCOMO estimation model:

$$Effort = a1 * (KLOC)^{a2}\ PM$$

$$T_{dev} = b1 * (efforts)^{b2}\ Months,$$

where KLOC is the estimated size of the software product indicate in Kilo Lines of Code, a1, a2, b1, b2 are constants for each group of software products, which were shown in the previouse table as a, b, c, d, respectively..

- And T_{dev} is the estimated time to develop the software, expressed in months.
- Effort is the total effort required to develop the software product, expressed in person months (PMs).

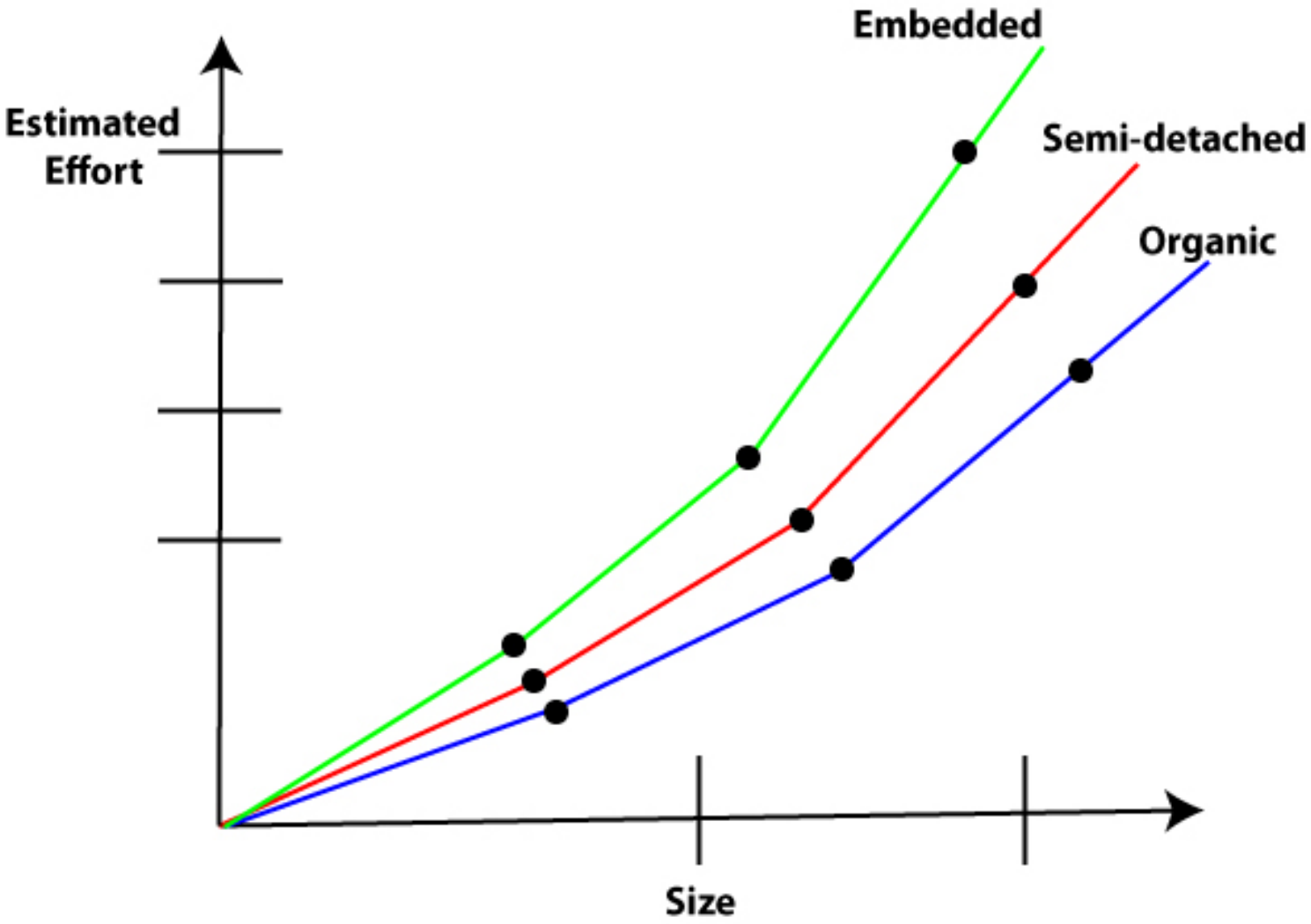
Estimation of development effort and time

- Estimation of development effort:
- For the three classes of software products, the formulas for estimating the effort based on the code size are shown below:
  - Organic: Effort $= 2.4(\text{KLOC})^{1.05}$ PM
  - Semi-detached: Effort $= 3.0(\text{KLOC})^{1.12}$ PM
  - Embedded: Effort $= 3.6(\text{KLOC})^{1.20}$ PM
- Estimation of development time:
- For the three classes of software products, the formulas for estimating the development time based on the effort are given below:
  - Organic: $T_{\text{dev}} = 2.5(\text{Effort})^{0.38}$ Months
  - Semi-detached: $T_{\text{dev}} = 2.5(\text{Effort})^{0.35}$ Months
  - Embedded: $T_{\text{dev}} = 2.5(\text{Effort})^{0.32}$ Months

Estimation of development effort and time

- Some insight into the basic COCOMO model can be obtained by plotting the estimated characteristics for different software sizes.
- The first figure (next slide) shows a plot of estimated effort versus product size.
- From figure, we can observe that the effort is somewhat superliner in the size of the software product.
- Thus, the effort required to develop a product increases very rapidly with project size.

Plot of estimated effort versus product size

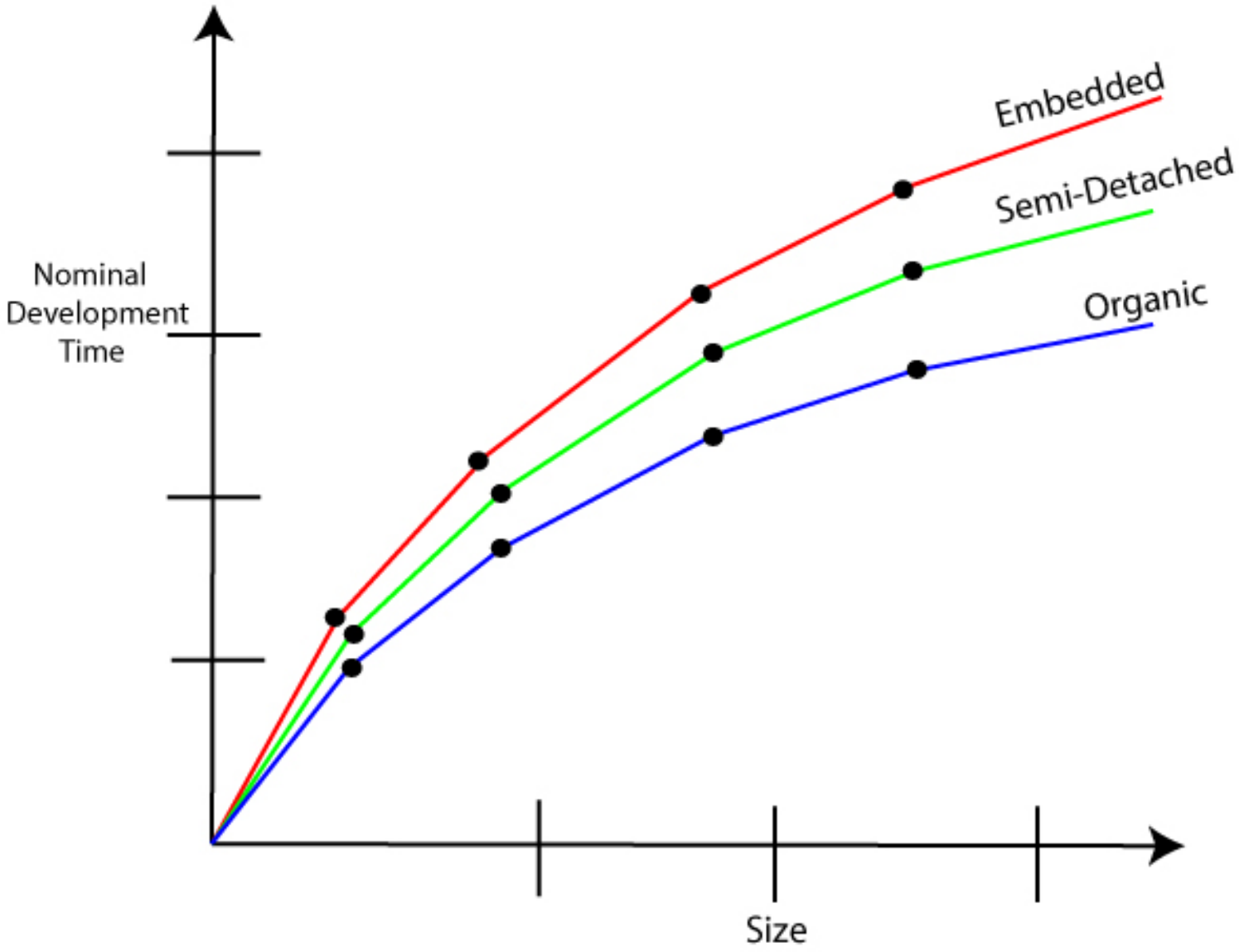


Effort versus product size

Plot of estimated effort versus product size

- The development time versus the product size in KLOC is plotted in fig.
- From fig it can be observed that the development time is a sub linear function of the size of the product, i.e. when the size of the product increases by two times, the time to develop the product does not double but rises moderately.
- This can be explained by the fact that for larger products, a larger number of activities which can be carried out concurrently can be identified.
- The parallel activities can be carried out simultaneously by the engineers.
- This reduces the time to complete the project.

Plot of the development time versus the product size



Development time versus size

Plot of the development time versus the product size

- Further, from the above figure, it can be observed that the development time is roughly the same for all three categories of products.
- **For example, a 60 KLOC program can be developed in approximately 18 months, regardless of whether it is of organic, semidetached, or embedded type.**
- See development versus size plot given in the previous slide, and focus on **the last three dots which are seen to be all aligned which corresponds actually to the same size (on the x-axis) corresponding to roughly the same development time (on the y-axis).**.

Plot of the development time versus the product size

- From the effort estimation, the project cost can be obtained by multiplying the required effort by the manpower cost per month.
- But, implicit in this project cost computation is the assumption that the entire project cost is incurred on account of the manpower cost alone.
- In addition to manpower cost, a project would incur costs due to hardware and software required for the project and the company overheads for administration, office space, etc.

Plot of the development time versus the product size

- It is important to note that the effort and the duration estimations obtained using the COCOMO model are called a nominal effort estimate and nominal duration estimate.
- The term nominal implies that if anyone tries to complete the project in a time shorter than the estimated duration, then the cost will increase drastically.
- But, if anyone completes the project over a longer period of time than the estimated, then there is almost no decrease in the estimated cost value.

Example 1: basic COCOMO model

Suppose a project was estimated to be 400 KLOC. Calculate the effort and development time for each of the three model i.e., organic, semi-detached & embedded.

- Solution: The basic COCOMO equation takes the form:

$$Effort = a1 * (KLOC)^{a2}\ PM$$

$$T_{dev} = b1 * (efforts)^{b2}\ Months,$$

- We are given that the Estimated Size of project = 400 KLOC.

  (i) Organic Mode:

$$E = Effort = 2.4 * (400)^{1.05} = 1295.31\ PM$$

$$D = T_{dev} = 2.5 * (1295.31)^{0.38} = 38.07\ PM$$

Solution: basic COCOMO model (cont.)

(ii) Semidetached Mode:

$$E = 3.0 * (400)^{1.12} = 2462.79\ PM$$

$$D = 2.5 * (2462.79)^{0.35} = 38.45\ PM$$

(iii) Embedded Mode:

$$E = 3.6 * (400)^{1.20} = 4772.81\ PM$$

$$D = 2.5 * (4772.8)^{0.32} = 38\ PM$$

Example 2: basic COCOMO model

A project size of 200 KLOC is to be developed. Software development team has average experience on similar type of projects. The project schedule is not very tight. Calculate the Effort, development time, average staff size, and productivity of the project.

- Solution: The semidetached mode is the most appropriate mode, keeping in view the size, schedule and experience of development time.
- Hence

$$Effort = E = 3.0 * (200)^{1.12} = 1133.12\ PM$$

$$T_{dev} = D = 2.5 * (1133.12)^{0.35} = 29.3\ PM$$

$$Average\ Staff\ Size(SS) = \frac{E}{D}\ Person = \frac{1133.12}{29.3} = 38.67\ Persons$$

$$Productivity = P = \frac{KLOC}{E} = \frac{200}{1133.12} = 0.1765 KLOC/PM = 176\ \frac{LOC}{PM}.$$

What is Risk?

- In simple terms, risk is the possibility of something bad happening.
- **"Tomorrow's problems are today's risk."**
- Hence, a clear definition of a "risk" is a problem that could cause some loss or threaten the progress of the project, but which has not happened yet.
- These potential issues might harm cost, schedule or technical success of the project and the quality of our software device, or project team morale.

Risk Management

- Risk Management is the system of identifying addressing and eliminating these problems before they can damage the project.
- A software project can be concerned with a large variety of risks.
- In order to be adept to systematically identify the significant risks which might affect a software project, it is essential to classify risks into different classes.
- The project manager can then check which risks from each class are relevant to the project.
- There are three main classifications of risks which can affect a software project:
  1. Project risks
  2. Technical risks
  3. Business risks

Risk Management: Project risks

- Project risks concern differ forms of budgetary, schedule, personnel, resource, and customer-related problems.
- A vital project risk is schedule slippage.
- Since the software is intangible, it is very tough to monitor and control a software project.
- It is very tough to control something which cannot be identified. For any manufacturing program, such as the manufacturing of cars, the plan executive can recognize the product taking shape.

Risk Management: Technical risks

- Technical risks concern potential method, implementation, interfacing, testing, and maintenance issue.
- It also consists of an ambiguous specification, incomplete specification, changing specification, technical uncertainty, and technical obsolescence.
- Most technical risks appear due to the development team's insufficient knowledge about the project.

Risk Management: Business risks

This type of risks contain risks of building an excellent product that no one need, losing budgetary or personnel commitments, etc.

Risk Management

1.Known risks: Those risks that can be uncovered after careful assessment of the project program, the business and technical environment in which the plan is being developed, and more reliable data sources (e.g., unrealistic delivery date)

2. Predictable risks: Those risks that are hypothesized from previous project experience (e.g., past turnover)

3. Unpredictable risks: Those risks that can and do occur, but are extremely tough to identify in advance.

Principle of Risk Management

1. **Global Perspective:** In this, we review the bigger system description, design, and implementation. We look at the chance and the impact the risk is going to have.
2. **Take a forward-looking view:** Consider the threat which may appear in the future and create future plans for directing the next events.
3. **Open Communication:** This is to allow the free flow of communications between the client and the team members so that they have certainty about the risks.
4. **Integrated management:** In this method risk management is made an integral part of project management.
5. **Continuous process:** In this phase, the risks are tracked continuously throughout the risk management paradigm.

3.2: Risk Management Activities

Risk management consists of two main activities which are each sub-divided into three activities, as shown in the figure below:

3.2.1: Risk Assessment

- The objective of risk assessment is to division the risks in the condition of their loss, causing potential.
- For risk assessment, first, every risk should be rated in two methods:
  - The possibility of a risk coming true (denoted as r).
  - The consequence of the issues relates to that risk (denoted as s).

  Based on these two methods, the priority of each risk can be estimated:

$$p = r * s,$$

  where p is the priority with which the risk must be controlled, r is the probability of the risk becoming true, and s is the severity of loss caused due to the risk becoming true.
- If all identified risks are set up, then the most likely and damaging risks can be controlled first, and more comprehensive risk abatement methods can be designed for these risks.

Risk Assessment: Risk Identification

1. Risk Identification: The project organizer needs to anticipate the risk in the project as early as possible so that the impact of risk can be reduced by making effective risk management planning.

A project can be of use by a large variety of risk. To identify the significant risk, this might affect a project. It is necessary to categories into the different risk of classes.

There are different types of risks which can affect a software project:

- Technology risks: Risks that assume from the software or hardware technologies that are used to develop the system. People risks: Risks that are connected with the person in the development team.
- Organizational risks: Risks that assume from the organizational environment where the software is being developed.

Risk Assessment: Risk Identification (cont.)

- Tools risks: Risks that assume from the software tools and other support software used to create the system.
- Requirement risks: Risks that assume from the changes to the customer requirement and the process of managing the requirements change.
- Estimation risks: Risks that assume from the management estimates of the resources required to build the system

Risk Assessment: Risk Analysis

2. Risk Analysis: During the risk analysis process, you have to consider every identified risk and make a perception of the probability and seriousness of that risk.

- There is no simple way to do this. You have to rely on your perception and experience of previous projects and the problems that arise in them.
- It is not possible to make an exact, the numerical estimate of the probability and seriousness of each risk. Instead, you should authorize the risk to one of several bands:
  1. The probability of the risk might be determined as very low (0-10%), low (10-25%), moderate (25-50%), high (50-75%) or very high (+75%).
  2. The effect of the risk might be determined as catastrophic (threaten the survival of the plan), serious (would cause significant delays), tolerable (delays are within allowed contingency), or insignificant.

3.2.2: Risk Control

- It is the process of managing risks to achieve desired outcomes.
- After all, the identified risks of a plan are determined; the project must be made to include the most harmful and the most likely risks.
- Different risks need different containment methods.
- In fact, most risks need ingenuity on the part of the project manager in tackling the risk.

Risk control: Planning methods in managing risks

There are three main methods to plan for risk management:

- **Avoid the risk:** This may take several ways such as discussing with the client to change the requirements to decrease the scope of the work, giving incentives to the engineers to avoid the risk of human resources turnover, etc.
- **Transfer the risk:** This method involves getting the risky element developed by a third party, buying insurance cover, etc.
- **Risk reduction:** This means planning method to include the loss due to risk. For instance, if there is a risk that some key personnel might leave, new recruitment can be planned.

Risk Leverage

- To choose between the various methods of handling risk, the project plan must consider the amount of controlling the risk and the corresponding reduction of risk.
- For this, the risk leverage of the various risks can be estimated.
- Risk leverage is the variation in risk exposure divided by the amount of reducing the risk.

Risk leverage = (risk exposure before reduction - risk exposure after reduction) / (cost of reduction)

1. Risk planning:

- The risk planning method considers each of the key risks that have been identified and develop ways to maintain these risks.
- For each of the risks, you have to think of the behavior that you may take to minimize the disruption to the plan if the issue identified in the risk occurs.
- You also should think about data that you might need to collect while monitoring the plan so that issues can be anticipated.
- Again, there is no easy process that can be followed for contingency planning. It rely on the judgment and experience of the project manager.

2. Risk Monitoring:

- Risk monitoring is an ongoing process of managing risks.
- Risk monitoring often has an initial phase which involves identifying risk, agreeing to the treatments and designing control.
- It is a process of tracking risk management execution and continuing to identify and manage new risks.
- It is the method of making sure that your assumption about the product, process, and business risks has not changed.

Project Scheduling

- Project-task scheduling is a significant project planning activity.
- It comprises deciding which functions would be taken up when.
- To schedule the project plan, a software project manager wants to do the following:
  1. Identify all the functions required to complete the project.
  2. Break down large functions into small activities.
  3. Determine the dependency among various activities.
  4. Establish the most likely size for the time duration required to complete the activities.
  5. Allocate resources to activities.
  6. Plan the beginning and ending dates for different activities. Determine the critical path.
  7. A critical way is the group of activities that decide the duration of the project.

Project Scheduling (cont.)

- The first method in scheduling a software plan involves identifying all the functions required to complete the project.
- A good judgment of the intricacies of the project and the development process helps the supervisor to identify the critical role of the project effectively.
- Next, the large functions are broken down into a valid set of small activities which would be assigned to various engineers.

Project Scheduling (cont.)

- The work breakdown structure formalism supports the manager to breakdown the function systematically after the project manager has broken down the purpose and constructs the work breakdown structure; he has to find the dependency among the activities.
- Dependency among the various activities determines the order in which the various events would be carried out.

Project Scheduling (cont.)

- If an activity A is necessary for obtaining the results of another activity B, then activity A must be scheduled after activity B.
- In general, the function dependencies describe a partial ordering among functions, i.e., each service may precede a subset of other functions, but some functions might not have any precedence ordering describe between them (called concurrent function).
- The dependency among the activities is defined in the pattern of an activity network.

Project Scheduling (cont.)

- Once the activity network representation has been processed out, resources are allocated to every activity.
- Resource allocation is usually done using a Gantt chart.
- After resource allocation is completed, a PERT chart representation is developed.
- The PERT chart representation is useful for program monitoring and control.

Project Scheduling (cont.)

- For task scheduling, the project plan needs to decompose the project functions into a set of activities.
- The time frame when every activity is to be performed is to be determined. The end of every action is called a milestone.
- The project manager tracks the function of a project by audit the timely completion of the milestones.
- If he examines that the milestones start getting delayed, then he has to handle the activities carefully so that the complete deadline can still be met.

Personnel Planning

- Personnel Planning deals with staffing. Staffing deals with the appoint personnel for the position that is identified by the organizational structure.
- It involves:
  - Defining requirement for personnel
  - Recruiting (identifying, interviewing, and selecting candidates)
  - Compensating
  - Developing and promoting agent
- For personnel planning and scheduling, it is helpful to have efforts and schedule size for the subsystems and necessary component in the system.

Personnel Planning (cont.)

- At planning time, when the system method has not been completed, the planner can only think to know about the large subsystems in the system and possibly the major modules in these subsystems.
- Once the project plan is estimated, and the effort and schedule of various phases and functions are known, staff requirements can be achieved.
- From the cost and overall duration of the projects, the average staff size for the projects can be determined by dividing the total efforts (in person-months) by the whole project duration (in months).
- Typically the staff required for the project is small during requirement and design, the maximum during implementation and testing, and drops again during the last stage of integration and testing.

Personnel Planning (cont.)

- Using the COCOMO model, average staff requirement for various phases can be calculated as the effort and schedule for each method are known.
- When the schedule and average staff level for every action are well-known, the overall personnel allocation for the project can be planned.
- This plan will indicate how many people will be required for different activities at different times for the duration of the project.
- The total effort for each month and the total effort for each step can easily be calculated from this plan.

Team Structure

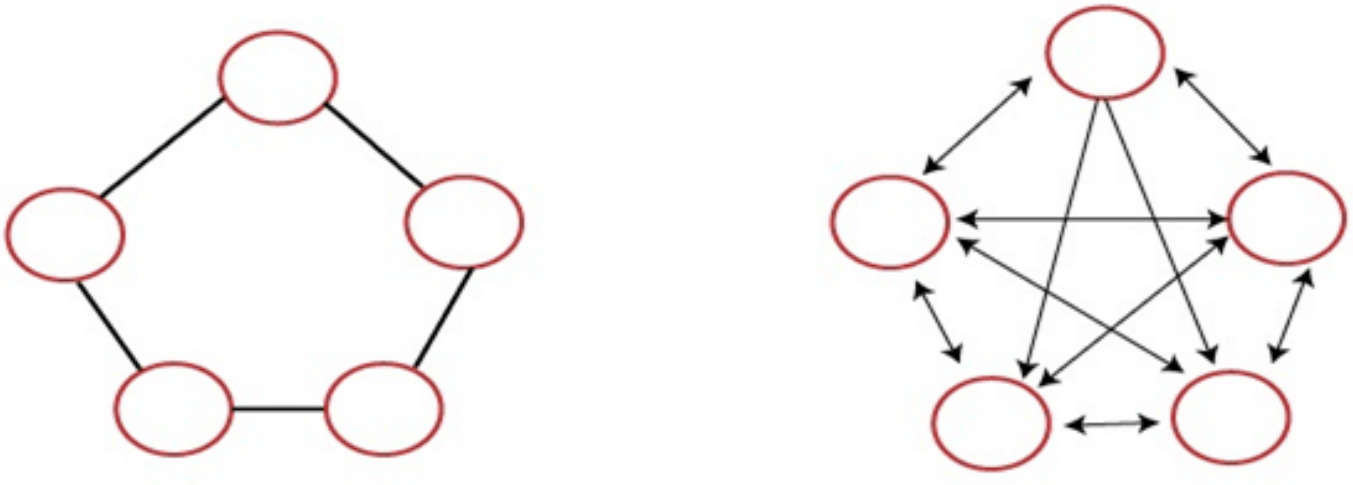
- Team structure addresses the issue of arrangement of the individual project teams.
- There are some possible methods in which the different project teams can be organized.
- There are primarily three formal team structures: **chief programmer, Ego-less or democratic, and the mixed team organizations** even several other variations to these structures are possible.
- Problems of various complexities and sizes often need different team structures for the chief solution.

Ego-Less or Democratic Teams

- Ego-Less teams subsist of a team of fewer programmers.
- The objective of the group is set by consensus, and input from each member is taken for significant decisions.
- Group leadership revolves among the group members. Due to its nature, egoless teams are consistently known as democratic teams.
- The structure allows input from all representatives, which can lead to better decisions in various problems.
- This method is well suited for long-term research-type projects that do not have time constraints.

Ego-Less or Democratic Teams

Ego-Less Programming Team structure and communication paths



(a) Structure

(b) Communication path

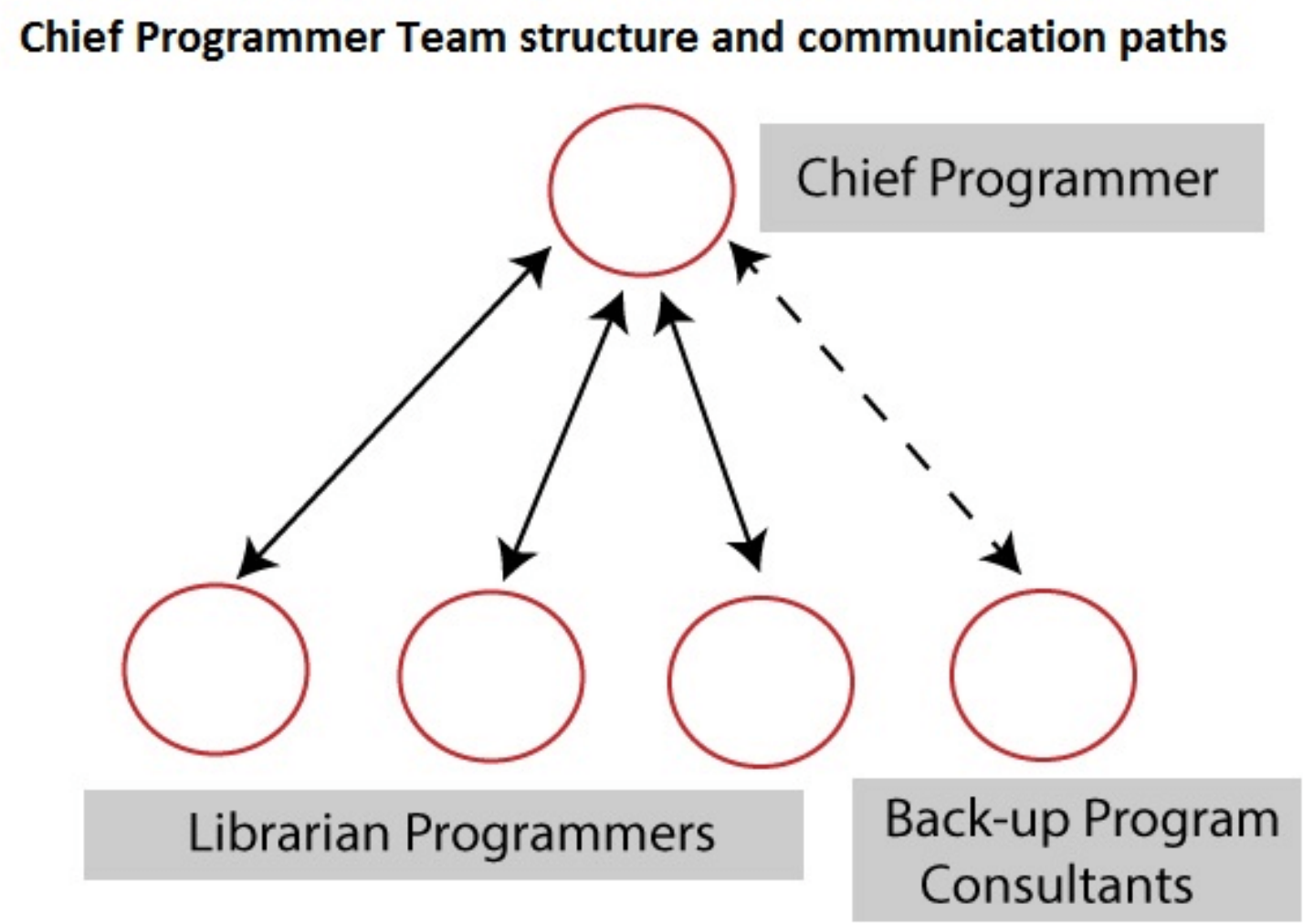
Chief Programmer Team

- A chief-programmer team, in contrast to the ego-less team, has a hierarchy. It consists of a chief-programmer, who has a backup programmer, a program librarian, and some programmers.
- The chief programmer is essential for all major technical decisions of the project.
- He does most of the designs, and he assigns coding of the different part of the design to the programmers.

Chief Programmer Team(cont's)

- The backup programmer uses the chief programmer makes technical decisions, and takes over the chief programmer if the chief programmer drops sick or leaves.
- The program librarian is vital for maintaining the documentation and other communication-related work.
- This structure considerably reduces interpersonal communication.

Chief Programmer Team(cont's)



Controlled Decentralized Team

- A third team structure known as the controlled decentralized team tries to combine the strength of the democratic and chief programmer teams.
- It consists of project leaders who have a class of senior programmers under him, while under every senior programmer is a group of a junior programmer.
- The group of a senior programmer and his junior programmers behave like an ego-less team, but communication among different groups occurs only through the senior programmers of the group.

Controlled Decentralized Team(cont's)

- The senior programmer also communicates with the project leader.
- Such a team has fewer communication paths than a democratic team but more paths compared to a chief programmer team.
- This structure works best for large projects that are reasonably straightforward. It is not well suited for simple projects or research-type projects.

Hierarchical Team Structure