CSC 408F/CSC2105F Lecture Notes

These lecture notes are provided for the personal use of students taking CSC 408H/CSC 2105H in the Fall term 2004/2005 at the University of Toronto.

Risk Management

- A *risk* is an event that causes something to go wrong on a project.
- Three characteristics of risks
  - If the risk happens, you lose something (e.g. time, money)
  - There is some non-zero likelihood that the risk will occur

    i.e. don't worry about meteors crashing into your project office.

    Risks with a probability of 1.0 of occurring are *problems* which must be dealt with.

    [*Prof. Dan Berry's story ...*]
  - The degree to which we can minimize the impact of the risk.
- *Risk control* attempts to minimize the probability of risks occurring and/or minimize its effect on the project.
- *Risk Exposure* (in dollars) is the sum $\sum_{i=1}^{i=maxRisks} Pr(risk_i) \cdot Cost(risk_i)$

  [*Any thing wrong with the formula?*]
- Management tracks risk exposure during a project and applies risk control to reduce exposure.

Generic Risks

- Lack of Common Understanding, ontology
  - Misunderstanding the requirements. *Clarify your requirements specifications*
  - Misunderstanding of target environment. *What is your target environment?*
  - Disputes among project teams.
- Bad planning
  - Inadequate time for testing
  - Misestimation of project difficulty or complexity
- Mishappening
  - Personnel turnover, loss of key people. *[How to deal with this problem?]*

Project Specific (Technical) Risks

- Failure of research component to converge to successful solution.
- Failure of sub-contractor to deliver promised hardware/software on schedule.
- Inadequate budget
- Unreasonable schedule
- Inadequate personnel or personnel organization
- Vague, incomplete, inconsistent requirements
- Inadequate resources allocated to project
- Uncooperative customer

Business (Social) Risks

- No one wants the product
- The organization doesn’t want the product
- The organization doesn’t know how to sell the product
- Senior management loses interest in the project
- Loss of budget or personnel allocated to the project
- Reorganization that makes project a non-core activity.
- Crisis point of the business [Andrew S. Grove. Only the Paranoid Survive: How to Exploit the Crisis Points That Challenge Every Company and Career.]
  Story: [Pentium Processor flaw] Story: [CISC vs RISC]

Boehm's Top Ten Risk Items[a]

1.	Personnel shortfalls	inexperience, turnover, team size
2,	Unrealistic schedules, budgets	bad estimates, inadequate resources
3.	Wrong functionality	wrong requirements, inadequate domain knowledge
4.	Wrong user interface	
5.	Gold plating	features not included in the requirements
6.	Requirements volatility	requirements changes during development
7.	Bad external components	software from others
8.	Bad external tasks	work performed by others
9.	Real-time performance shortfalls	software too slow for requirements
10.	Capability shortfalls	unstable environment, untried technology

[a]van Vliet Figure 8.3

Risk Management

- Identify risk factors $risk_i$
- Estimate risk probability $Pr(\ risk_i\ )$

  Estimate risk effect (cost) $Cost(\ risk_i\ )$
- Calculate total risk exposure

$$Risk\ Exposure = \sum_{i=1}^{i=maxRisks} Pr(\ risk_i\ ) \cdot Cost(\ risk_i\ )$$

- Develop strategies to mitigate the risks.

  Usually only N risks with highest $P(\ risk_i\ )$ [*The 90-10 or 80-20 rule]*
- Handle risks that occur.
- Re-evaluate risk assessment periodically.

Risk Estimation (Projection)

- Risk projection activities
  1. Identify likelihood of each perceived risk
  2. Determine the consequences of the risk
  3. Estimate impact of risk on project and product
  4. Determine uncertainty bounds for risk estimates
- Impact estimation
  1. *Nature* of the problem associated with the risk
  2. *Scope* - how serious is risk and how much of project will be affected
  3. *Timing* - will the risk arise early or late in the project?

Risk Mitigation Strategies

- *Risk Avoidance*

  Change requirements so risk no longer applies.

  Example: delete functionality that looks hard to implement.

  Example: give project more resources, improve staff training.

- *Risk Transfer*

  Change development process (e.g. do prototyping)

  Make your risk someone else's risk.

  Examples: Project completion insurance, move risk to client.

- *Risk Assumption*

  Accept potential risk and it's consequences as part of the project.

  Take preventative actions to reduce probability and impact.

  Prepare contingency plan to deal with the risk if it does occur.

Practical Risk Management

- Identify risks that *really* matter
- Do whatever is feasible and affordable to prevent the risk or to be prepared for the risk
- Be able to detect if the risk has occurred
- Have a *contingency plan* in place to deal with the risk if it does occur
- Update risk estimates and contingency plans as project moves ahead.
- All this risk management takes resources and may add delays to the project. The amount of risk management actually done depends on the magnitude of the project and the consequences of the expected risks

Risk Management Example

- **Risk** - Projected high staff turnover
- *Pre-project Prevention*
  - Try to address the causes of the high turnover
    Fix any problems that can be fixed (e.g. working conditions)
  - Assume high turnover will occur. Structure project to compensate for the effects of high turnover.
- *Intra-project Compensation*
  - Organize project teams so information about the project is widely disseminated
  - Demand detailed documentation in all phases of the project.
    Monitor to ensure that documentation is kept current
  - Use peer reviews (inspections, walkthroughs) of all work as a mechanism to disseminate project knowledge
  - Identify critical personnel and assign a backup person to each. The backup person should keep current with the critical persons work. *[Extreme Programming, Extreme -ing]*

Risk Documentation

- Risk Description
  - Statement of Risk – condition and consequences
  - Context – circumstances, environment, other issues
  - Impact – affected products, schedules, activities
  - Time frame – period during which risk is real.

    Period during which action can/should be taken.
  - Probability of risk occurring.
  - Mitigation strategy - action to eliminate, reduce or prevent the risk
- Risk Management and Control Information
  - Status and Priority (high, medium, low)
  - Date opened/identified. Risk origin (who identified the risk)
  - Assignment (person responsible for analysis and mitigation strategy.
  - Status/date (status changes, e.g. change is probability, related events)
  - Date closed (risk no longer relevant)

People Management and Team Organization

- All large software development is done by *teams*
- Organizing the available personnel into effective and efficient teams is one challenging aspect of software project management.
- People management issues
  - Assignment of people to teams/tasks.
  - Coordination of the activities of teams/individuals.
  - Enabling communication within teams and between teams.
  - Monitoring the performance and progress of teams and individuals.
  - Maintaining the perspective that developers are people not machines.

People Management

- Teams are made up of *individuals*.

  Each individual has a personal set of goals and abilities.

- Project management has to meld groups of individuals into teams.

  The goals of each individual must be reconciled into the goals of the project as a whole. *[When a project involves mutliple teams ...]*

- It is important to
  - Identify the goals of the project early.
  - Clearly communicate those goals to the teams and individuals.

- Team members must have a clear understanding of what is expected of them.

- Teams with uncertain or ambiguous goals will invent their own goals which are usually not the desired project goals.

- Project Goals are the standard against which project progress and individual performance is evaluated.
- Examples of project goals
  - High level of correctness.
  - Completion by a specified date.
  - Compatability with other software systems.
  - Meet specific performance standards.
  - Maintainability.
- Need to be careful about metrics and techniques used to measure individual and team progress.

  Example: *lines-of-code per month* metric encourages wasteful programming style and discourages software reuse. *[Same applies to documentations ...]*

  Example: *use of errors found during reviews* as a programmer metric discourages frank and open discussion of software problems. Antagonizes developers. *[Our teams must cooperate to gain higher marks!]*

Types of Coordination Mechanisms[a]

- *Direct Supervision - Simple Structure*
  - A few managers supervise a small group of workers in a loosely defined simple structure.
  - Typical of small and new organizations.
  - Low specialization, training and formalization.
  - Coordination depends on efforts of individuals
- *Standardization of Work Processes - Machine Bureaucracy*
  - Work to be done is completely specified and susceptible to automated production.
  - Low specialization, training, formalization.
  - Coordination via standardized process, e.g. assembly line.

[a] Mintzberg, *Designing Effective Organizations*

- *Standardization of Work Outputs - Divisionalized*
  - Individual divisions (projects or subprojects) have considerable autonomy as to the methods used to achieve the project goals.
  - Coordinations is achieved through standardization of work outputs, e.g. implementing a module to a strictly defined interface.
  - Control is exerted by periodic assessment of progress for each division.
- *Standardization of Worker Skills - Professional Bureaucracy*
  - Skilled programmers are given freedom to do their best job on particular parts of the system.
  - Coordination through interchangeability of workers, uniformity of tools and processes used by the individuals.
  - Requires considerable *trust* that the skilled workers will get the job done correctly and efficiently.

- *Adhocracy*
  - Suitable for large projects with high uncertainty, high innovation component.
  - Many specialists cooperate in development of the software.
    Each specialist contributes according to their own expertise.
    Example: brainstorming.
  - Specialists have great latitude to find a solution for their assigned part of the project.
    Project success depends on the specialists successfully converging to an ill-specified goal in some non-specified way.
  - Coordination is achieved through *mutual adjustment*
    Example: negotiating the definition of an interface between two parts of the software.
    Example: negotiating with the client about the functionality of the system.

Management Directions

- *Relation directness* - attention to each individual and their relationship to other individuals in the organization.
- *Task directedness* - attention to the results to be achieved and the way in which the results must be achieved.
- A manager can separately emphasize or de-emphasize each of these directions in their management of the project.

Management Style	Relation Directness	Task Directness
Separation	low	low
Commitment	low	high
Relation	high	low
Integration	high	high

Separation Management Style

- Similar to Mintzberg's *coordination through standard work processes.*
- Most effective for routine work.
- Management acts bureaucratically and specifies work rules and procedures.
- Efficiency is the central theme.
- Work coordination is hierarchical.
  Decision making is top-down, formal, authority based.
- Advantage: stable, well-defined project organization.
- Disadvantage: bureaucracy and hierarchy make truly innovative results unlikely.

Commitment Management Style

- Similar to Mintzberg's *professional bureaucracy*
- Most effective for work under pressure (eg. deadlines)
- Manager provides the vision and the solution.
- Decision making based on a shared vision of the desired outcome.
- Disadvantage: once the *true path to success* has been identified, hard to change direction or emphasis.

Relation Management Style

- Similar to Mintzbergs's *mutual adjustment coordination mechanism.*
- Most effective when people need to be motivated, coordinated and trained.
- Tasks are assigned to individuals. Work is not routine but requires innovation by the individual.
- Decision making is a group process, by negotiation and consensus building.
- Disadvantages: high level of interaction on decisions requires time and effort. Success depends on manager's ability to achieve consensus among project members.

Integration Management Style

- Similar to Mintzberg's *mutual adjustment coordination mechanism.*
- Used when the result is uncertain.
- Work is explorative, experimental. High interdependence between tasks.
- Managers role is to stimulate and motivate personnel.
- Decision making is informal, bottom-up.
- Advantages: promotes creativity, encourages individual excellence.
- Disadvantage: goals of individuals may conflict with project goals causing distortions.

Human Resources

- Organizing and managing people is one of the most important parts of project management
- There are many ways to organize people in a software project
  - *Sweatshop Mode* - A large number of programmers are assigned individual programming tasks (i.e. a procedure). Several levels of managers are responsible for coordinating the activities of these programmers
  - *Assembly Line Mode* - Programmers work in informal teams to implement a collection of related programming tasks (i.e. a module) Managers coordinate the activities of the teams
  - *Team Mode* - Programmers work in formally defined teams with intra-team management and coordination. Coordination is controlled by the team and by software managers

Matching People to Tasks

- There are many different kinds of activities in a project

  analysis, design, implementation, testing, documentation, etc.

- Different individuals will be effective and efficient at different kinds of activities.

- The performance of individual programmers may vary by more than an order of magnitude. Examples

Programming Speed	1:10	1 day	vs.	1.5 weeks
Code Size	1:25	100 KB	vs.	2.5 MB
Debugging time	1:15	1 hour	vs.	2 days

- We don’t have good methods for estimating parameters of programmer performance with respect to the activities in a project.

Personnel Capabilities

- Ability to perform the work
- Interest in the work (project)
- Experience with similar projects
- Experience with project tools and languages
- Experience with similar techniques
- Experience with the project development environment
- Education and training
- Ability to communicate with others
- Written communication skills
- Ability to share responsibility with others
- Management skills

[Self-assessment, 360 degree assessment]

Personnel Assignment

- A critical part of project management is the assignment of the available personnel to the activities in the project.
- Try to simultaneously optimize several dimensions
  - Assign each individual to the activities where they will be most effective and efficient.
  - Assign individuals so that they will work well together and cooperate on shared activities.
  - Assign the ”best” people to the critical paths.
  - Try to keep the personnel motivated and productive.

Ego-less Programming[a]

- Weinberg's work has a major impact on the way programmers are managed
- Ego-less programming - programmers should be encouraged to suppress their own interests (ego) for the good of the team and the project
- All team members should feel equally responsible for the success of the project
- Readers and writer - an early form of peer review to encourage cooperation among team members
- Weinberg emphasizes importance of setting correct goals for the team

[a] G.M. Weinberg, *The Psychology of Computer Programming*, Van Nostrand Reinhold, 1971

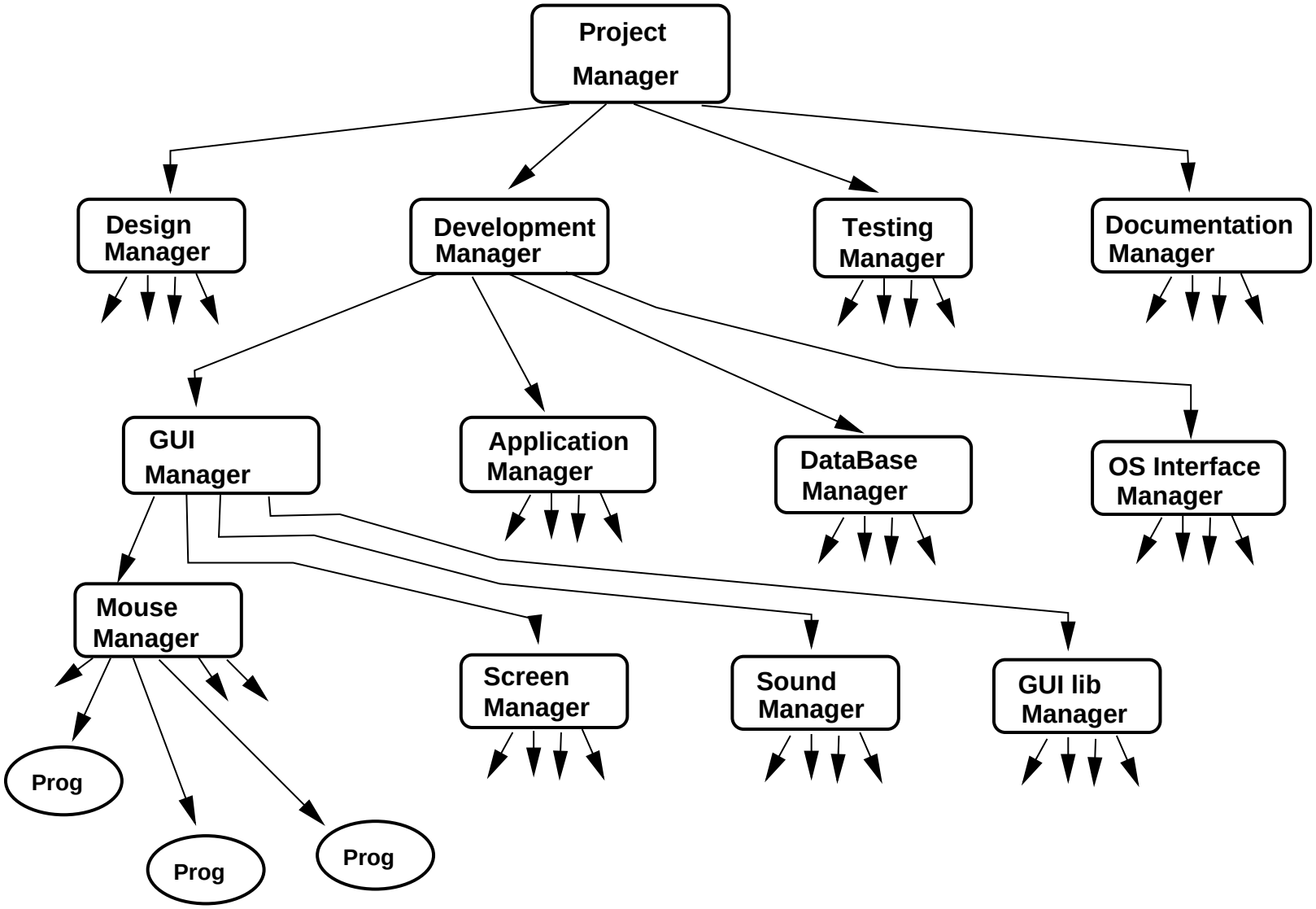
Ways to Organize Personnel

- Organizational Strategies
  - Hierarchical Tree
  - Informal Team
  - Chief Programmer Teams
  - Matrix organization
  - SWAT Team
- Issues
  - A project size increases, more structure is needed.
  - Informal teams cope with high uncertainty better.
  - **Efficient communication is a key issue**
  - Don’t stifle creativity with too much structure.
  - Software structure often mirrors organization structure.

Hierarchical Organization

- A tree-like management structure with the real work being done at the leaves (developers). Interior nodes are various levels of management.
- The lowest level of managers control the work of one team.
  Higher level managers coordinate the work of several teams.
- Can (probably will) use different management styles in different parts of the hierarchy.
- Disadvantages
  - Communication follows the tree structure unless specific steps are taken to facilitate communication.
  - Specific knowledge tends to dissipate as it moves up the tree.
  - Decisions propagate downward in the tree, may be distorted before they reach the leaves.

Hierarchical Project Example



Matrix Organization

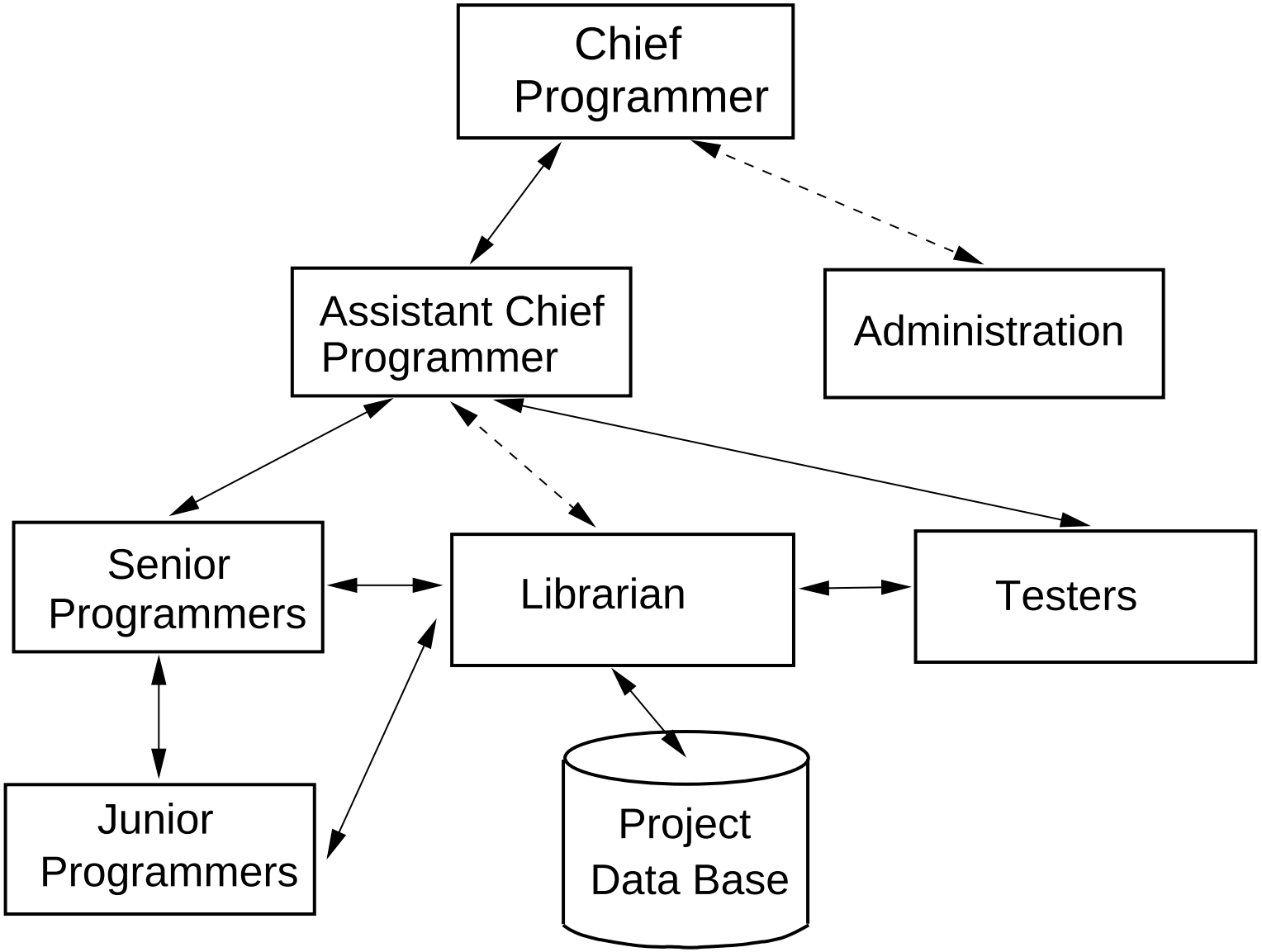
- Identify people with particular skill sets.
  Assign those people to projects needing those particular skills.
- An individual may be simultaneously involved in several projects reporting to different managers.
- Could also do this with small teams instead of individuals.
- Manager is responsible for coordination and overall direction.
- Works well if the sharing of people works well and managers cooperate.
  Makes effective use of skilled personnel/teams.
- Disadvantages: working for multiple bosses is never easy.

Chief Programmer Teams[a]

- Clever organization of a programming team modeled on a hospital surgical team. Goals: increase productivity of programmers and quality of software
- The key ideas
  - Let each person concentrate on doing those tasks that they can do best
  - Librarian does *all* clerical tasks: program editing, compilations, maintaining archives, running canned tests, etc.
  - Chief Programmer concentrates on *technical issues* not management
  - Simplifies communication and reporting structure.
- Issues
  - Scalability to large projects?
  - Librarian still necessary/useful?

[a] F.T. Baker, *Chief Programmer Team Management of Production Programming*, IBM Systems Journal, 1972

Chief Programmer Team Example



SWAT Team[a]

- Small team of highly skilled developers. Generalists not specialists.
- Works cooperatively and informally on project tasks.
  Environment organized for efficient informal communication.
- Uses brainstorming, workshops and prototyping to develop system design and implementation.
- Team uses incremental development, software reuse, very-high level programming languages, software generators to maximize productivity.
  Use groupware software *[E.g. OmniEditor]* to coordinate/manage team activities.
- SWAT teams are usually highly motivated and tightly knit.

[a] A squad of policemen who have been trained to deal with violent and dangerous situations

HOW TO Organize a Team

- Use fewer better people.
- Try to fit the tasks to the capabilities and motivations of the people available.
- Help people get the most out of themselves.

  Encourage education, professional development and advancement.
- Select people that work well together for the same team.

  Well balanced and harmonious.
- Pro-actively deal with disruptive, dysfunctional team members.

  If a person doesn’t work well in a given team, management should do something about it.

Stenning's Project Hygiene
Introduction

It is suggested that many of the difficulties encountered by systems and software projects are not the result of deep technical problems, but rather arise from a lack of *basic project hygiene* - from failure to enforce various elementary principles and disciplines that are self evidently prerequisite to a successful outcome. Such disciplines are primarily concerned with the control and co-ordination of both project *activities* and project *products* [a]

[a] Vic Stenning, *Project Hygiene*,Proceedings Usenix Software Management Workshop, new Orleans, 1989

[a] Stenning slides ©David Tilbrook, 1997

Stenning's Project Hygiene Principles

1. Everyone involved in the project should know the objectives of what he or she is doing.
2. Achievement of the overall project objectives should follow immediately from the achievement of all individual objectives.
3. Both individual objectives and overall project objectives should be realistic.
4. There should be a known method for addressing each individual object.
5. Changes should be controlled, visible and of known scope.
6. Both people and products should be insulated from the effects of changes that are not (currently) of relevance.

Stenning's Project Hygiene Suggestions

1. Focus on the Process as a whole rather than on the (final) product.
2. Invest more effort in ”higher level” descriptions
3. Coordinate activities as well as products.
4. Adopt a strict policy on project phases.
5. Provide more semantic information to the configuration management and build systems.

Stenning's Project Hygiene
Fundamental Questions

- Are the objectives of all project activities clearly and precisely defined?
- Are there any activities for which the objectives are completely unrealistic?
- How will the results of the individual activities be combined to meet the objectives of the overall project?
- Are there explicit procedures and mechanisms to ensure traceability?
- Are there effective mechanisms for coordinating changes and limiting the scope of their impact?

Reading Assignment

van Vliet Chapter 5, 6