Unit 9: Project Management and Project Management Tools

Er. Rajan Karmacharya
Department of CSIT
St. Xavier's College,
Maitighar

Includes…

1. Software configuration management
2. SCM tasks and roles
3. Risk Management
4. Risk Management Process
5. SPM Tools

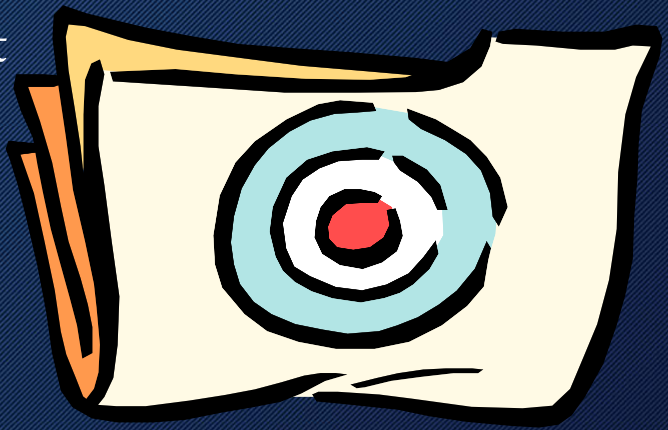
Software Configuration Management

Software Configuration Management (SCM)

The First Law of System Engineering

“ No matter where you are in the system life cycle, the system will change, and the desire to change it will persist throughout the life cycle.”

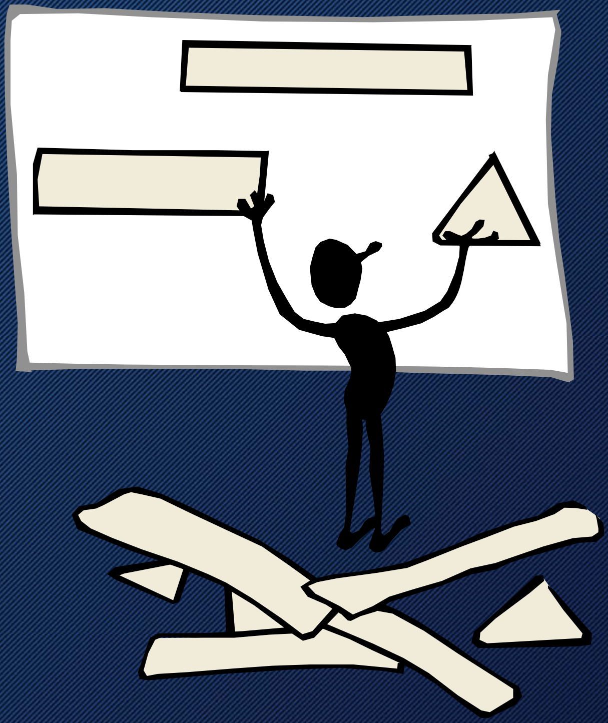
- Ideal
  - Software is developed from stable/frozen requirements
  - The concept is that it is easier to hit a stationary target than a moving target
- Reality
  - Not applicable for most real-world systems
- The only constant is “CHANGE”
  - An effective software project need to have a strategy to tackle “CHANGE”



How software changes….

- The four aspects of software evolution are:

1. Corrective changes
2. Adaptive changes
3. Perfective changes
4. Preventive changes



Corrective Changes

- Required to maintain control over the system's day-to-day functions
- These changes are made as faults (or) bugs are found during the development time
- Some changes may be long-term and fundamental, some may be patches to keep the system in operation (emergency fixes)

Adaptive Changes

- Essentially maintaining control over system modifications
- As one part of the system changes, other impacted areas will need to be updated
- Examples
  - Database upgrades
  - Use of a new compiler or development tool

Perfective Changes

- Perfecting existing acceptable functions
- The domain of Refactoring designs falls into this category
- Perfective changes are done to increase the long-term maintainability or elegance of the solution
    - Involves changes to design or data structures for better efficiency
    - Updates to documentation to improve their quality
    - Enhancing the code to make it more readable

Preventive Changes

- Preventing the system performance from degrading to unacceptable levels
- Involves alterations made to ensure that the system has a defense against potential failures
- Example:
  - Adding extra redundancy modules to ensure that all transactions are properly logged

Types of Changes

- The typical distribution of these changes is (from Lientz & Swanson 1981):
  - Perfective (50%)
  - Adaptive (25%)
  - Corrective (21%)
  - Preventive (4%)
- These figures will change depending on the system and project

Changes and Control

- If changes are not controlled in a project – things can and will get out of hand
- The issue of change management is even more important when multiple people work on a project as well as on the same deliverable
- Without proper strategies and mechanisms to control changes – one can never revert back to an older more stable copy of the software
  - Important as every change introduces risk into the project

So what is the answer??

- The facts:
  - Change is unavoidable in software
  - Changes needs to be controlled
  - Changes need to be managed
- The solution
  - Software configuration management (SCM)

New Version

Significant change in functionality, technology, hardware and software requirements

New Release

Only a bug fix, minor enhancement in functionality

Configuration Management…

- This is the discipline that applies a rigorous approach to ensure
  - Different items produced in software systems are all identified and tracked
  - Changes to the various items are recorded and tracked
  - Completion and proper integration of all the various modules
- SCM can help determine the impact of change as well as control parallel development
- It can track and control changes in all aspects of software development
  - Requirements
  - Design
  - Code
  - Tests
  - Documentation

Need for SCM…

- As software evolves – many resources make changes to the system
  - CM prevents avoidable errors that arise from conflicting changes
- Often many versions of the software are released and require support
  - CM allows a team to support many versions.
  - CM allows changes in sequential versions to be propagated
- CM allows developers to track changes and reverse any fatal changes to take a software system back to its last known safe state

Need for SCM…

- Good SCM increases confidence that we are:
  - Building the right system
  - Testing the system enough
  - Changing it correctly and carefully
- It also:
  - Restrains non-essential changes
  - Ensures that decisions and changes are traceable
  - Increases accountability
  - Improves overall software quality
  - Provides a fall back position when things do not work

Significance of SCM

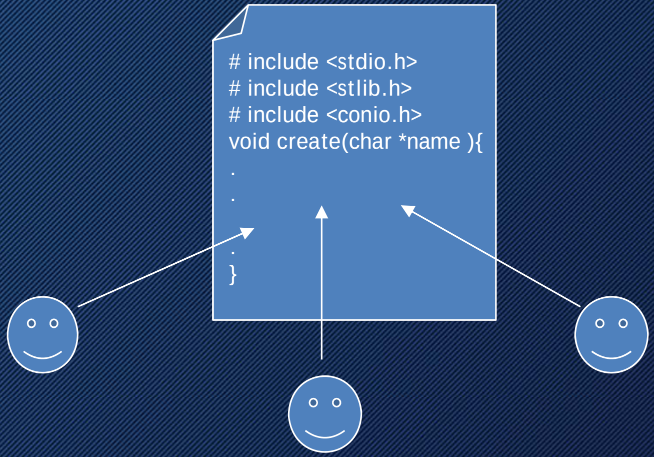
Because change can occur at any time, SCM activities are developed to

- Identify Change
- Control Change
- Ensure that change is being properly implemented
- Report changes to others who may have an interest.
- Control access to deliverables of software project

The need of SCM

Concurrent Access:

- Single copy of program
- Many working on it
- Carry out changes simultaneously
- Overwrite each other while saving



```
# include <stdio.h>
# include <stlib.h>
# include <conio.h>
void create(char *name ){
.
.
.
}
```

SCM Activities

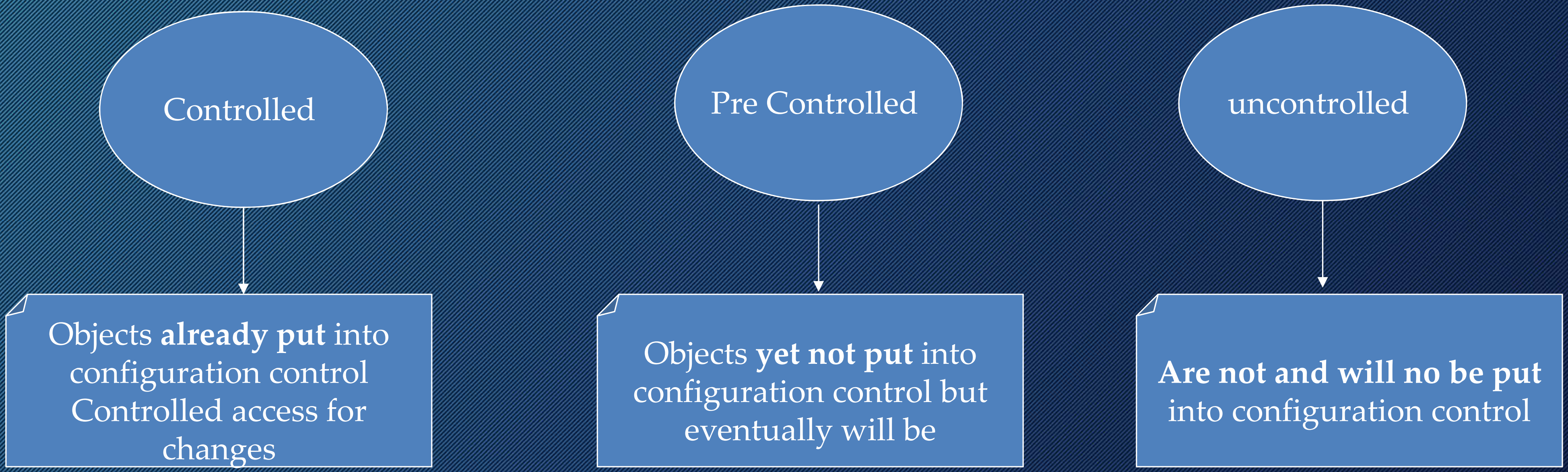
i. **Configuration Identification**

Which part of system to be kept record of ?

ii. **Configuration Control**

Ensures changes to a system happens smoothly!!

Configuration Identification



Controlled

Objects **already put** into configuration control Controlled access for changes

Pre Controlled

Objects **yet not put** into configuration control but eventually will be

uncontrolled

Are not and will no be put into configuration control

Configuration Control

- Process of managing changes to controlled objects
- Prevents unauthorized changes to any controlled object

A Baseline

A specification **or product that has been** formally reviewed and agreed upon**, that thereafter** serves as the basis **for further development, and that can be changed only through** formal change control **procedures!!**

- Before a software configuration item becomes a baseline, change may be made quick and informal.
- However, once a baseline is established, we figuratively pass through a swinging one way door.
- Changes can be made, but a specific, formal procedure must be applied to evaluate and verify each change.

SCM Roles and Responsibilities…

- Configuration manager
  - Responsible for approving configuration items
  - Responsible for development and enforcement of procedures
  - Approves STM (ship to manufacture) level release
  - Responsible for monitoring entropy
- Change control board
  - Approves and prioritizes, or rejects change requests
- Software engineers
  - Responsible for identification and versioning of configuration items
  - Create promotions triggered by change requests or the normal activities of development.
  - Update the items to incorporate requested changes – they also resolve any merge conflicts

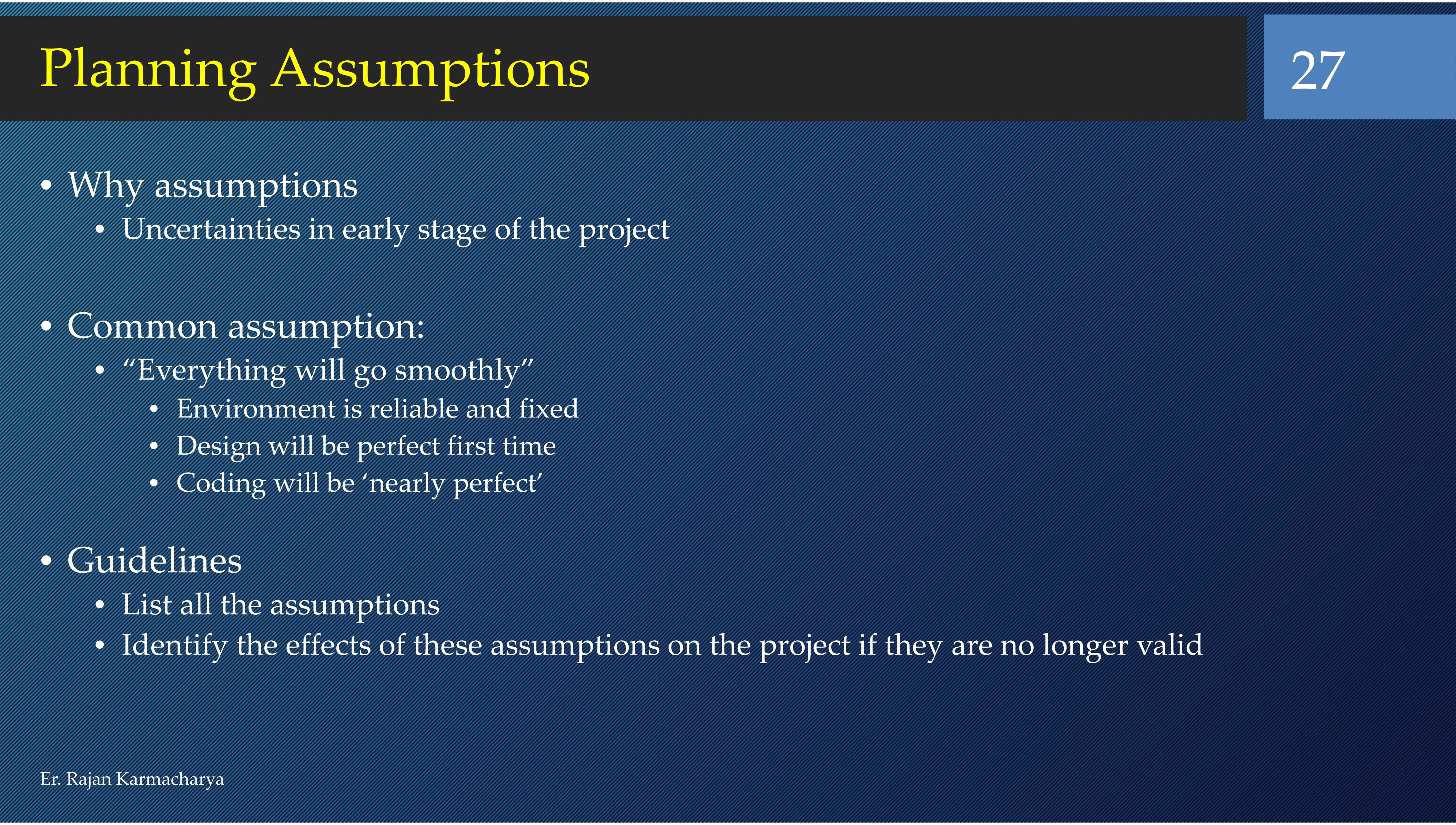
Risk Management

Project Risks

- Factors that cause a project to be delayed or over-budget

Nature of Project Risks

- Planning assumptions
- Estimation errors
- Eventualities (Possibilities)



Planning Assumptions

- Why assumptions
  - Uncertainties in early stage of the project
- Common assumption:
  - “Everything will go smoothly”
    - Environment is reliable and fixed
    - Design will be perfect first time
    - Coding will be ‘nearly perfect’
- Guidelines
  - List all the assumptions
  - Identify the effects of these assumptions on the project if they are no longer valid

Estimation Errors

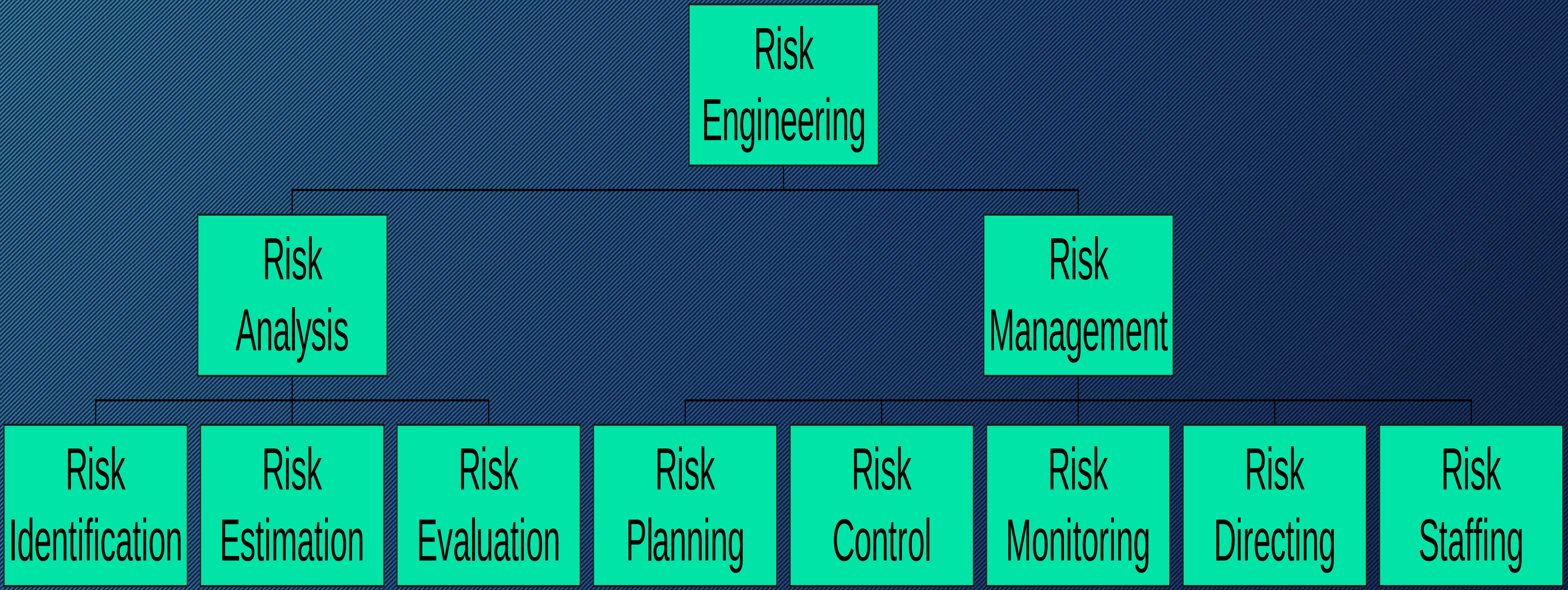
- Difficult to have accurate size or time estimations
  - Lack of experience of similar tasks
  - Lack of historical data
  - Nature of the task

- Estimation can be improved by analyzing historic data for similar tasks and similar projects
  - Keep historic data of your estimation and the actual performance
  - Compare your estimation and the actual value
  - Classify the tasks that are easy or difficult to give accurate estimation

Eventualities

- Unexpected and unimaginable events
- Common unexpected events
  - Hardware cannot be delivered on time
  - Requirements specification needs to be rewritten
  - Staffing problem

Boehm's Risk Engineering



Risk Engineering

Risk Analysis

Risk Management

Risk Identification

Risk Estimation

Risk Evaluation

Risk Planning

Risk Control

Risk Monitoring

Risk Directing

Risk Staffing

Risk Identification

- Identify the hazards that might affect the duration or resource costs of the project

Hazard → Problem → Risk

- A *hazard* is an event that might occur and will create a problem for the successful completion of the project, if it does occur

Hazard, Problem, and Risk

- *Hazard*: Mary's baby may be born early
- *Problem*: Modules P and Q will have no coder
- *Risk*: Milestone 7 will be delayed, or extra budget will be needed to hire another coder

Risk Identification (cont'd)

- Type of risks
  - *Generic risk* (common to all projects)
    - Standard checklist can be modified based on the risk analysis of previous projects
  - *Specific risk* (only applies to individual projects)
    - More difficult to find
    - Need to involve project team members
    - Need an environment that encourages risk assessment

Risk Identification (cont’d)

- Guideline
  - Use checklist that lists the potential hazards and their corresponding factors
  - Maintain an updated checklist for future projects

Common Risk Factors

- Application factors
- Staff factors
- Project factors
- Hardware and software factors
- Changeover factors
- Supplier factors
- Environment factors
- Health and safety factors

Application Factors

- Nature of the application
  - A data processing application or a life-critical system (e.g. X-ray emission system)
- Expected size of the application
  - The larger is the size, the higher is the chance of errors, communication problems and management problems

Staff Factors

- Experience and skills
- Appropriateness of experience
- Staff satisfaction
- Staff turn-over rates

Project Factors

- Project objectives:
  - Ill defined
  - Unclear to every team member and user
- Project methods:
  - Ill specified methods
  - Unstructured methods

Hardware and Software Factors

- New hardware
  - Stability of the new hardware system??
- Cross platform development
  - Development platform is not the operation platform
  - Does the language used support cross platform development?

Changeover Factors

- ‘All-in-one’ changeover
  - The new system is put into operation
- Incremental or gradual changeover
  - Adding new components to the system by phases
- Parallel changeover
  - Both the existing system and the new system are used in parallel

Supplier Factors

- Late delivery of hardware
- Instability of hardware
- Late completion of building sites

Environment Factors

- Changes in environment such as hardware platforms
- Changes in government policies
- Changes in business rules
- Restructuring of organizations

Health and Safety Factors

- Health and safety of staff and environment
  - Staff sickness, death, pregnancy etc.
  - Any tragic accident to staff

Boehm's Top Ten Risk Items

- Personnel shortfalls
- Unrealistic schedules and budgets
- Developing the wrong software functions
- Developing the wrong user interface
- Gold plating
- Continuing stream of requirements changes
- Shortfalls in externally performed tasks
- Shortfalls in externally furnished components
- Real-time performance shortfalls
- Straining computer science capabilities

Risk Management

- Risk planning
- Risk control
- Risk monitoring
- Risk directing
- Risk staffing

Risk Planning

- Making contingency plans
- Where appropriate, adding these plans into the project's overall task structure

Risk Control

- Minimizing and reacting to problems arising from risks throughout the project

Risk Monitoring

- It is an ongoing activity throughout the whole project to monitor
  - the likelihood of a hazard; and
  - the impact of the problem caused.

Risk Directing and Staffing

- These concerns with the day-to-day management of risk.
- Risk aversion strategies and problem solving strategies frequently involve the use of additional staff and this must be planned for and should be considered.

Risk Reduction Strategies

- 5 different types in a generic sense
  - Hazard prevention
  - Likelihood reduction
  - Risk avoidance
  - Risk transfer
  - Contingency planning

Hazard prevention

- Prevent a hazard from occurring or reduce its likelihood to an insignificant level
    - Lack of skilled staff can be prevented by employing staff with appropriate skills
    - Unclear requirements specification can be prevented by using formal specification techniques

Likelihood reduction

- Reduce the likelihood of an unavoidable risk by prior planning
    - Late change to the requirements specification can be reduced by using prototyping

Risk avoidance

- Some hazards cannot be avoided but their risks may
  - A project can be protected from the risk of overrunning the schedule by increasing duration estimates.

Risk transfer

- The impact of the risk can be transferred away from the project by contracting out or taking out insurance
  - The risk of shortfalls in external supplied components can be transferred away by quality assurance procedures and certification, and contractual agreements.

Contingency planning

- Contingency plans are needed to reduce the impact of those risks that cannot be avoided
    - The impact of any unplanned absence of programming staff can be minimized by using agency programmers

Any Queries?

ST. XAVIER'S COLLEGE

Department Of Computer Science
and Information Technology

Er. Rajan Karmacharya