



TOGETHER
for a sustainable future

OCCASION

This publication has been made available to the public on the occasion of the 50th anniversary of the United Nations Industrial Development Organisation.



TOGETHER
for a sustainable future

DISCLAIMER

This document has been produced without formal United Nations editing. The designations employed and the presentation of the material in this document do not imply the expression of any opinion whatsoever on the part of the Secretariat of the United Nations Industrial Development Organization (UNIDO) concerning the legal status of any country, territory, city or area or of its authorities, or concerning the delimitation of its frontiers or boundaries, or its economic system or degree of development. Designations such as “developed”, “industrialized” and “developing” are intended for statistical convenience and do not necessarily express a judgment about the stage reached by a particular country or area in the development process. Mention of firm names or commercial products does not constitute an endorsement by UNIDO.

FAIR USE POLICY

Any part of this publication may be quoted and referenced for educational and research purposes without additional permission from UNIDO. However, those who make use of quoting and referencing this publication are requested to follow the Fair Use Policy of giving due credit to UNIDO.

CONTACT

Please contact publications@unido.org for further information concerning UNIDO publications.

For more information about UNIDO, please visit us at www.unido.org

22693

**UNIDO Training Workshop Report
On Software Development and Management
From 19 to 30 November 2001**

I. Background

1. Subject

Promoting IT popularization and exchange ---- the 2001 IT Training Activities in the Asia-Pacific Region.

2. Theme

Management and technical development of the software programs

3. Bases

According to the project document *Enhancing IT Cooperation and Partnerships in the Asia-Pacific Region* signed on May 25 2001 by Shanghai Municipal Government and UNIDO on the Second High-Level Forum on City Informatization in the Asia-Pacific Region, we shall actively support human resource development, especially the training of IT personnel in the less developed countries.

4. Target

In order to promote city informatization in the Asia-Pacific region, especially in the developing countries in this region; to exchange the latest technology in software technology and software project management; to improve the level of management and technology of the software industry in the developing countries

5. Time

From Nov.19th, 2001 to Nov.30th, 2001

6. Courses

There are 2 courses in the training workshop:

- (1). CMM software project management;
- (2). Software development process using UML

II. Training Work

With the kind support of UNIDO and Shanghai Government .The training activities were successful carried out from Nov.19, 2001 to Nov.30, 2001. In order to popularize the training activities, UNIDO actively invited those participants recommended by their own governments, while having provided round-trip air tickets to some participants from the developing countries. The Shanghai Municipal Government whose attention was paid to these training activities, actively organized, finalized personnel, ground, facilities and carefully arranged the students' daily life

during this training.

1.Enrolment

The enrolment of participants began from September 1, 2001 and ended on October 15, 2001. During this period we made full use of multi-channels to actively launch the publicity for the training workshop while having overcome the unfavorable factors of international situation. After careful consideration and selection, 20 participants including the government officers, researchers, engineers, project managers with software project background coming from 11 countries in the Asia-Pacific Region had participated in the training.

Our training workshop has received energetic support from the Foreign Affairs Office of the Shanghai Municipal Government, who rendered the visa service to the participants, and ensured the participants' smooth entry.

2.Service

The workshop has two courses, i.e. CMM software project management and Software development process using UML. Every course is further divided into 10 teaching units, totaling 40 teaching hours.

Professor He Jiefeng, senior fellow of the United Nations university and Dr. Rajest Vasa from a U.S. famous IT company were invited to give lectures in Shanghai. Their lectures were focused on the technical practices and application as well as engineering experience, which is of great significance for reference.

According to the project memorandum signal between the UNIDO and RCOCI (UNIDO Project No.2001/300) the collection, compilation and editing materials for CMM and UML training courses were successfully completed within two months. These materials were timely printed and given to the workshop for us.

The corresponding teaching facilities and places, for instance, computers, projectors, conference rooms and hall, motor rooms and library, etc, were provided while management personnel for educational administration who offered convenience to participants' daily life during the training.

Given considerations to the insecure factors arising from the international situation, we had bought accident insurance for all participants.

3.Action

Shanghai Municipal Government offered the kind support to this training workshop. Fan Xiping, the Director and He Shouchang, the Deputy Director of the Shanghai Municipal Informatization Office attended the opening ceremony.

The participants were organized to visit the Shanghai Super Computer Center where they could directly experience the fruit of city informatization construction.

Two seminars were arranged for the participants to probe into the policies and laws or regulations in the cyberspace while exchanging the successful experiences and solution so participants own countries.

The participants are arranged to visit Shanghai International Industrialization Fair, especially to see about the IT Production Hall of the Fair.

The Participants have visited the Bund and the Shanghai Museum, as well as places of cultural interest, which enriched students' extracurricular life.

YANG Xiong, the Vice Secretary General of Shanghai Municipal Government attended the graduation ceremony and issued the certificates to the students. Then,

evening party was held with a brilliant performance given by the participants themselves, which had left deep impression to everybody.

III. Training Fruits

The training activities had obtained the anticipated results through the vivid and brilliant lectures given by teachers and through practical discussions and case studies.

1. Through the learning of the management curriculums of software program, the trainees could fully:

- Understand the essence and characteristics of software project, the international standard ISO 9000, as well as the Capability Maturity Model CMM;
- Select suitable software process model for the supervision of the software project;
- Apply suitable software to prepare the planning of software project;
- Apply suitable software to assess the scale and cost of the software project;
- Understand the technology for the management of software project so as to provide products with good quality and less risk;

2. After the study of the UML curriculum the participants came to understand the problems emerged from the large-scale software development while solving these problems with face object technology. In understanding and using the facing-object technology to analyze, standardize, design and realize the super computer system, now the trainees could:

- Analyze the demands of customers;
- Use UML to supply standards for proper scale system;
- Design on the basis of object oriented standards;
- Realize designing and clearly grasp the activities carried out and products produced in the software development;
- Comprehensively master the basic principle and technology of the software development with Object Oriented Data Technology.

IV. Participants' Suggestions

The profound friendship between the participants and also between the participants and teacher were established through 12 days studies. A fellow-participants association was initiated to maintain the contacts while sharing the information and focusing the attention to the technical development. All participants reached consensus that the training activities are of great significance, from which they gained a great deal of enlightenment and experience for the future works. Therefore, they do hope to attend such as kind of workshop in the near future. Some participants suggested that a workshop on policies and laws or regulations in the cyberspace should be held. Thus, the participants can have more time to discuss and exchange the successful experiences of the IT technology, IT industry and its application so as to promote the informatization construction in every country. In the meantime, they also expressed gratitude to the organizers of the workshop, which had provided a learning, exchanging and cooperation opportunity to them.

V. Conclusion

With the development of information technology and coming of networked new economy, the software industry is becoming the core of 21st century and even national economy development. The significance of introducing CMM & UML into the training course is aimed improving the understanding of software development and the management, providing a great amount of program experience in practice while having successful management experience and discussed technical applications. And as a carrier, training class has provided an exchange and cooperation platform for exchanging situation and experience of informatization construction, so as to encourage the mutual understanding and cooperation while promoting the technical progress and the development of the city informatization.

The training activities have also signified the start of the establishment of the Shanghai International Center for Information Technology Promotion, which is a good beginning for future training activities. Shanghai will work hard for offering active, continuous and systemic training to trainers and managers of the city informatization in the Asia-Pacific Region, and for promoting progress in city informatization construction management and techniques and for narrowing the 'digital divide' of the developing countries in the Asia-Pacific Region and other regions.

The Regional Cooperation Office for City Informatization

Annex 1

The 2nd Session of Technical Training on Informatization

—Software Development and Management

Program Calendar

| | Sunday | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday |
|---------------------|--------------------------------------|--|---------|-------------------------------------|----------------------|------------------------|--|
| Date | 11/18 | 11/19 | 11/20 | 11/21 | 11/22 | 11/23 | 11/24 |
| Morning 8:30-- | Registration | The Beginning Ceremony | UML② | CMM② | UML④ | CMM④ | Shanghai One-day Tour: Museum, Art Museum, Grand Theatre |
| Afternoon 1:00-- | | UML① | CMM① | UML③ | CMM③ | UML⑤ | |
| Evening | | Welcome Buffet | | Sightseeing: Bund | | | |
| Date | 11.25 | 11.26 | 11.27 | 11.28 | 11.29 | 11.30 | 12.1 |
| Morning 8:30-- | UML⑥ | CMM⑤ | CMM⑥ | UML⑧ | UML⑨ | CMM⑨ | Check-out Transport to Airport |
| Afternoon 1:00-- | Shanghai International Industry Fair | Visit SSC, Science & Technology Museum | UML⑦ | CMM⑦ Seminar (Shanghai Info-office) | CMM⑧ Seminar (SMERT) | UML⑩ Graduate Ceremony | |
| Evening | | | | Shopping Tour | | Party | |

Annex 2

Training Workshop to Improve Software Engineering and Development

(From 19 to 30 November 2001 in Shanghai, China)

Participants List

| ID | Country | City | Name | Organization | Official Title | Gender |
|----|------------|-------------|----------------------------|---|--|--------|
| 1 | Bangladesh | Dhaka | Mohammad Mosharraf Hossain | Bangladesh Computer Council | Deputy Director | male |
| 2 | Bangladesh | Dhaka | Mohammad Sirajul Islam | Dhaka City Corporation | Urban Planning Department | male |
| 3 | Cameroon | Yaounde | Yvonne Elenda | Ministry of Finances | Engineer in Computer | female |
| 4 | China | Hefei | Dong Lanfang | University Of Science & Technology Of China | Department of Computer Science and Technology, | female |
| 5 | China | Hefei | Zhang Yu | University Of Science & Technology Of China | Department of Computer Science and Technology, | female |
| 6 | Indonesia | Jakarta | Gerson Damanik | Directorate General of Post and Telecommunication | Directorate Spectrum Frequency and Satellite Orbit | male |
| 7 | Laos | Vientiam | Chanhaboury Sengchanh | Electricite Du Laos | Deputy Manager Computer Department | male |
| 8 | Laos | Vientiam | Somphith Kedvichith | Ministry of Industry and Handicraft | Deputy chief of decision of Management | male |
| 9 | Mongolia | Ulaanbaatar | Battsengel Tseveenravdan | Bodicomputer Co., Ltd. | System & Software Engineer | male |

| ID | Country | City | Name | Organization | Official Title | Gender |
|----|-------------|-------------|-------------------------|--|--|--------|
| 10 | Mongolia | Ulaanbaatar | Battulga Bayabsaikhan | COMBYTE Co., Ltd. | software engineering | male |
| 11 | Mongolia | Ulaanbaatar | Delgersaikhan Dagvadorj | Mongolian University of Science and Technology Computer Science and Management School | Lecture | male |
| 12 | Nepal | Kathmandu | Pramod Pradhan | Kathmandu Valley Mapping Programme | System Administrator | male |
| 13 | Nepal | Kathmandu | Sarojani Joshi | Kathmandu Valley Mapping Programme | System Manager | female |
| 14 | Philippines | Quezon City | Herbert Langiden | League of Municipalities of the Philippines | Information and Communications Technology Staff | male |
| 15 | Syria | Damascus | Issam Badrich | State Planning Commission(SPC) | Head Section Energy & Mining Dir | male |
| 16 | Syria | Damascus | R'ed Jaum'a | State Planning Commission(SPC) | Head Section Energy & Mining Dir | male |
| 17 | Thailand | Bangkok | Ariya Wichitnuntakorn | National Electronics and Computer Technology Center(NECTEC) | Researcher | female |
| 18 | Thailand | Bangkok | Nongkran Chompoonut | National Electronics and Computer Technology Center(NECTEC) | Assistant Director | female |
| 19 | Vietnam | Hanoi | Pham Vam Phao | Ministry of Foreign Affairs | Engineer | male |
| 20 | Vietnam | Hanoi | Xuan Hoa Nghiem | Ministry of Foreign Affairs | Officer | male |

Annex 2

Training Workshop to Improve Software Engineering and Development
(From 19 to 30 November 2001 in Shanghai, China)

Lecture List

| ID | Country | City | Name | Organization | Official Title | Gender | Course |
|-----------|----------------|-------------|-------------|---|----------------------------|---------------|---------------|
| 1 | America | Portland | Rajesh Vasa | Webgain Inc | Senior Business Consultant | male | CMM |
| 2 | China | Macau | He Jifeng | International Institute of Software Technology United Nations University | Senior Research Fellow | male | UML |

Annex 2

Training Workshop to Improve Software Engineering and Development (From 19 to 30 November 2001 in Shanghai, China)

Participants List (The round-trip tickets are provided to these participants.)

| ID | Country | City | Name | Organization | Official Title | Gender |
|----|------------|----------|------------------------|-------------------------------------|---------------------------------------|--------|
| 1 | Vietnam | Hanoi | Pham Vam Phao | Ministry of Foreign Affairs | Engineer | male |
| 2 | Vietnam | Hanoi | Xuan Hoa Nghiem | Ministry of Foreign Affairs | Officer | male |
| 3 | Laos | Vientiam | Somphith Kedvichith | Ministry of Industry and Handicraft | Deputy chief of diction of Management | male |
| 4 | Bangladesh | Dhaka | Mohammad Sirajul Islam | Dhaka City Corporation | Urban Planing Department | male |



Software Project Management

Lecture 1---Introduction to Software Project Management

Lecture 2---Software Development Models

Lecture 3---Size and Cost Estimation

Lecture 4---Software Project Planning

Lecture 5---Software Project Risk Management

Lecture 6---Resource Allocation

Lecture 7---ISO 9000

Lecture 7A---SEI - Capability Maturity Model

Lecture 8---Software Project Performance Tracking and Monitoring

Lecture 9---Software Configuration Management

Lecture 10---Project Team Management & Organization

Software Project Management

Lecture 1 Introduction to Software Project Management

Overview

- Software engineering
- Software project management
- Formal methods

Software Project Management

2

Software Crisis

- Faulty software
- Delay in completion time
- Over budgeted
- Difficult to maintain software

Software Project Management

3

Some important facts

- Relative cost of the software in a system is growing
- Increase in demand for software
- Increase in size and complexity of software
- Increase in performance of hardware

Software Project Management

4

Software Engineering

- No standard definitions

Software Project Management

5

Software Engineering

- Aimed at large software
- Systematic and well-defined techniques, methodologies and tools
- To design, code, test and maintain quality software
- Within a resource constrained environment

Software Project Management

6

Large Software

- Developed by more than one person
- Effective communications are important – standards, documentations, etc
- Management issues
- Techniques and methodologies are useful only if automated systems can be built upon them

Software Project Management

7

Phases of Software Development

- Requirements analysis and specifications
- Design
- Coding
- Testing
- Operation
- Maintenance

Software Project Management

8

Some important observations

- Maintenance is the most expensive phase and coding is the least expensive phase
- The earlier the detection of faults, the less expensive the correction of faults

Software Project Management

9

Characteristics of software

- Simple and elegant mathematical representation
- Logic intensive
- Cannot have partial completion
- Design costs are more expensive

Software Project Management

10

What is a project?

- Key characteristics of a project:
 - A planned activity
 - Specific objectives or products
 - Work to be carried out in several phases
 - Limited resources
 - Deadline
 - Large and complex

Software Project Management

11

Major differences between software products and hardware products

- Progress of software development is not obviously visible
- Modifications of software products are more easy and flexible
- Software products are usually more complex than the hardware products in terms of development or construction cost

Software Project Management

12

Major processes in developing a software system

- Feasibility study
- Project planning
- Project execution

Feasibility Study

- Analyze the general requirements, costs and the functionalities and services provided by the system to be developed
- Aimed to determine whether a system should be developed or not
- Note that a feasibility study can be viewed as a project itself

Important factors in planning a software project

- To know the nature of the system to be developed
 - A management information system or a control system
- To know clearly the objectives and products of the project
 - How to evaluate the objectives and products after the completion of the project

What is management?

- Management involves the following activities:
 - Planning
 - Staffing
 - Innovating
 - Directing
 - Monitoring
 - Liaising

What is software project management?

- Understand the characteristics of software products
- Understand what is meant by project
- Understand what is meant by management

Common problems with software projects

- Lack of quality standards and measures
- Lack of measurable milestones
- Difficult to make the progress visible
- Poor communications
- Poor documentation
- Frequent changes of requirements
- Over budget and late delivery of software

Major issues of software project management to be covered

- Software development models
- Software size and cost estimation
- Software project planning
- Software risk management
- Resource allocation
- ISO 9000
- Performance tracking and report

Major issues of software project management to be covered (cont'd)

- Software project configuration management
- Software project team management

Main problems encountered with requirements and specifications

- Ambiguous
- Incomplete
- Inconsistent

Main problems encountered with requirements and specifications (cont'd)

- To overcome these problems via
 - Formality – achieving preciseness
 - Abstraction – contracting on essential parts

Formal Methods

- Mathematically based techniques
 - Providing a universal and concise language
 - Supporting formality
 - Supporting abstraction
 - Supporting logical reasoning
 - May support automation

Why formal methods?

- Unambiguous
- Concise
- May support automatic verification
- May support automatic processing

Why not formal methods?

- Lack of knowledge
- Lack of experienced staff
- Lack of supporting tools
- Cost-effectiveness

Formal Methods

- Most commonly used mathematical techniques:
 - Set theory
 - Logic

Formal Methods (continued)

- Two important aspects:
 - Expressiveness
 - Solvable properties – automation

Logic

- Propositional calculus
- First-order predicate calculus
- Second-order predicate calculus



Example (Expressiveness):

- Consider the following:
Every citizen has an ID number.
Since John is a citizen, John has an ID number.
This kind of information cannot be deduced using the propositional calculus, first-order predicate calculus is required.

Solvable Properties

- A problem is solvable if there is an algorithm which can determine "yes" or "no" as the solution to the problem
- A problem is unsolvable if it is not solvable.
- A problem is partially solvable if there is an algorithm which can determine "yes" as the solution to the problem

Solvable Properties (continued)

- The validity problem of the propositional calculus is solvable
- The validity problem of the first-order predicate calculus is unsolvable, but partially solvable.
- The validity problem of the second-order predicate calculus is not partially solvable.

A Dilemma

- More expressive
- Solvable in preference to partially solvable
- Partially solvable in preference to unsolvable

Z

- Developed at Oxford University (UK) in the late 1970s and early 1980s
- Use set theory and first-order logic to model the requirements
- Can check inconsistency among requirements

Z

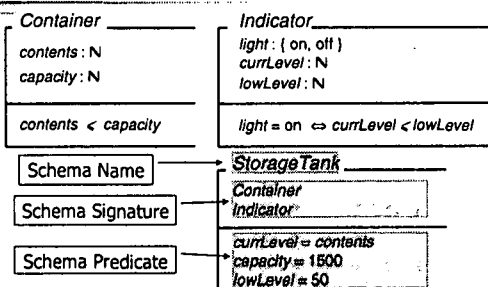
- A specification language not a programming language
 - A what-to not a how-to
- Specify pre-conditions and post-conditions
- Based on state-based models

Essentials of State-Based Model

Besides *variables* and their *types*, we have:

- State
- Invariant
- Input and output
- Operation

An Example: Container and Indicator



An Example: A Storage Tank

StorageTank

contents : N
 capacity : N
 light : { on, off }
 currLevel : N
 lowLevel : N

contents < capacity
 light = on \Leftrightarrow currLevel < lowLevel
 currLevel = contents
 capacity = 1500
 lowLevel = 50

An Example: Filling the Tank

Message ::= okMsg | overflowMsg

FillTank

Δ StorageTank
 fillAmount? : N
 msg! : Message

currLevel + fillAmount? < capacity
 currLevel' = currLevel + fillAmount?
 msg! = okMsg

Overflow

\exists StorageTank
 fillAmount? : N
 msg! : Message

currLevel + fillAmount? > capacity
 msg! = overflowMsg

DoFillTank \equiv FillTank \vee Overflow

References

- Hughes, B., and Cotterell, M. (1999) *Software Project Management*, 2nd ed., McGraw Hill
- Dean, C.N., and Hinchey, M.G. (1996) *Teaching and Learning Formal Methods*, Academic Press
- Sommerville, I. (2001) *Software Engineering*, 6th ed., Pearson Education.

Software Project Management

Lecture 2 Software Development Models

Overview

- Introduction
- Technical plan
- Software process models
- Selecting process model

Definitions

- Software Process
 - the set of activities, methods, and practices that are used in the production and evolution of software
- Software Process Model
 - one specific embodiment of a software process architecture

(Humphrey 1990)

Why Modelling?


- To provide a common understanding
- To locate any inconsistencies, redundancies and omissions
- To reflect the development goals and provide early evaluation
- To assist development team to understand any special situation

Project Analysis

- Methodologies
 - Object-Oriented Development (OOD)
 - Structured System Analysis and Design Method (SSADM)
 - Jackson Structured Programming (JSP)
- Technologies
 - application-building environment
 - knowledge-based system tools

Project Characteristics


- data oriented or control oriented system?
- general package or application specific?
- a particular type of system for which specific tools have been developed?
- safety-critical system?
- nature of the hardware/software environment?



Project Risks

- Product uncertainty
- Process uncertainty
- Resource uncertainty


Software project management 7



Considerations for Project Approach

- Control systems
- Information systems
- General applications
- Specialized techniques
- Hardware environment
- Safety-critical systems
- Imprecise requirements


Software project management 8



Technical Plan

- Contents
 - Constraints
 - Approach
 - Implementation
 - Implications


Software project management 9



Technical Plan - Constraints

- Character of the system to be developed
- Risks and uncertainties of the project
- User requirements concerning implementation


Software project management 10



Technical Plan - Approach

- Selected methodology or process model
- Development methods
- Required software tools
- Target hardware/software environment

Software project management 11



Technical Plan - Implementation

- Development environment
- Maintenance environment
- Training

Software project management 12

Technical Plan - Implications

- Project products and activities
 - effect on schedule duration and overall project effort
- Financial
 - report used to produce costings

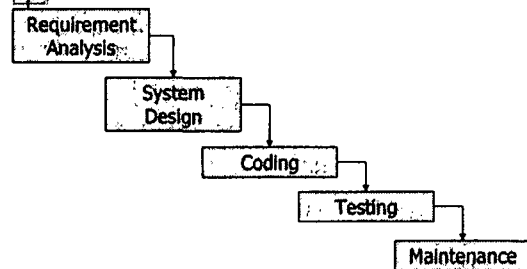
Software Process Models

- Waterfall Model
- V Model
- Spiral Model
- Prototyping Model

Software Process Models (cont'd)

- Phased Development Model
 - incremental development model
 - iterative development model
- Operational Specification Model
- Transformation Model

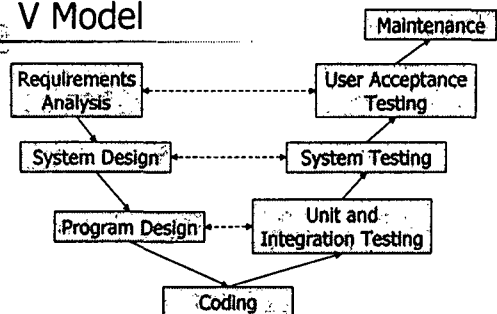
Waterfall Model



Waterfall Model (cont'd)

- classical
- one-shot approach
- effective control
- limited scope of iteration
- long cycle time
- not suitable for system of high uncertainty

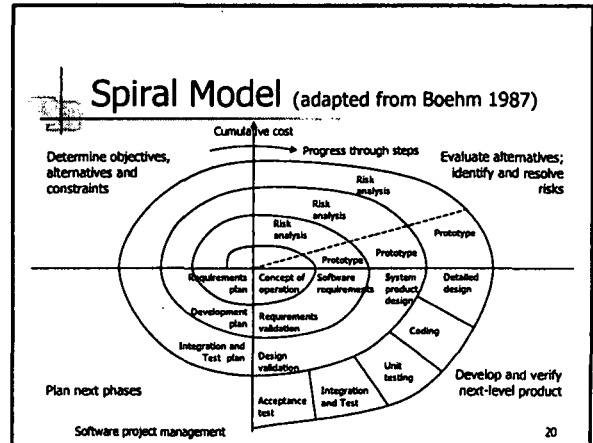
V Model



V Model (cont'd)

- Additional validation process introduced
- Relate testing to analysis and design
- Loop back in case of discrepancy

Software project management 19



Spiral Model (cont'd)

- Evolutionary approach
- Iterative development combined with risk management
- Risk analysis results in "go, no-go" decision

Software project management 21

Spiral Model (cont'd)

- Four major activities
 - Planning
 - Risk analysis
 - Engineering
 - Customer evaluation

Software project management 22

Prototyping Model

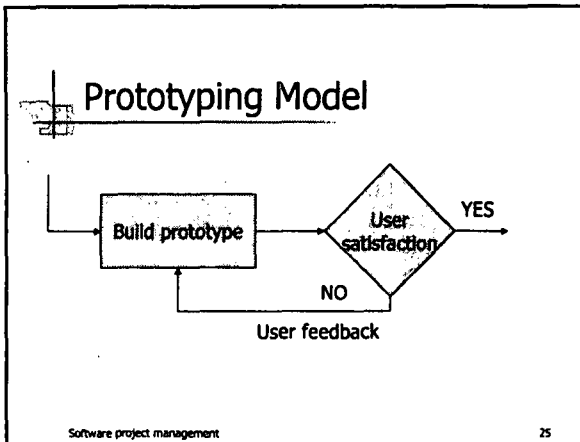
- Goals
 - meet users' requirements in early stage
 - reduce risk and uncertainty

Software project management 23

Classification of Prototype

- Throw-away
 - After users agree the requirements of the system, the prototype will be discarded.
- Evolutionary
 - Modifications are based on the existing prototype.
- Incremental
 - Functions will be arranged and built accordingly.

Software project management 24



- ### Benefits of Prototyping
- Learning by doing
 - Improved communication
 - Improved user involvement
 - Clarification of partially-known requirements
- Software project management 26

- ### Prototyping Sequences
- Requirements gathering
 - Quick design
 - Prototype construction
 - Customer evaluation
 - Refinement
 - Loop back to quick design for fine tuning
 - Product engineering
- Software project management 27

- ### Benefits of Prototyping
- Demonstration of the consistency and completeness of a specification
 - Reduced need for documentation
 - Reduced maintenance costs
 - Feature constraint
 - Production of expected results
- Software project management 28

- ### Drawbacks of Prototyping
- Users sometimes misunderstand the role of the prototype
 - Lack of project standards possible
 - Lack of control
 - Additional expense
 - Machine efficiency
 - Close proximity of developers
- Software project management 29

- ### Forms of Prototypes
- Mock-ups
 - Simulated interaction
 - Partial working model
- Software project management 30

Prototype Products

- Human-computer interface
- System functionality

Software project management 31

Prototype Changes

- Three categories
 - Cosmetic (35%)
 - screen layout
 - Local (60%)
 - screen processing
 - Global (5%)
 - multi-parts processing
 - design review

Software project management 32

Phased Development

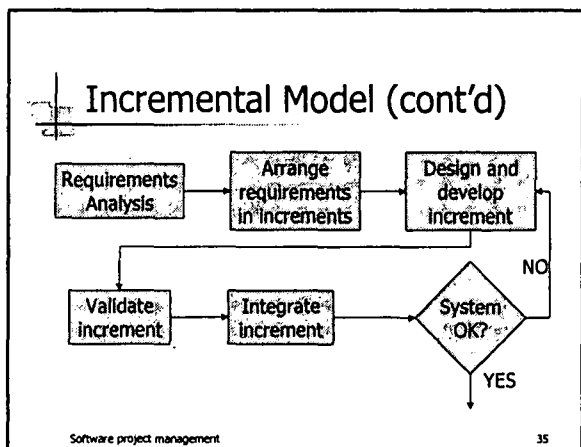
- Reduce cycle time
- Two parallel systems:
 - operational system (Release n)
 - development system (Release n+1)
- Two approaches
 - incremental
 - iterative

Software project management 33

Incremental Model

- Break system into small components
- Implement and deliver small components in sequence
- Every delivered component provides extra functionality to user

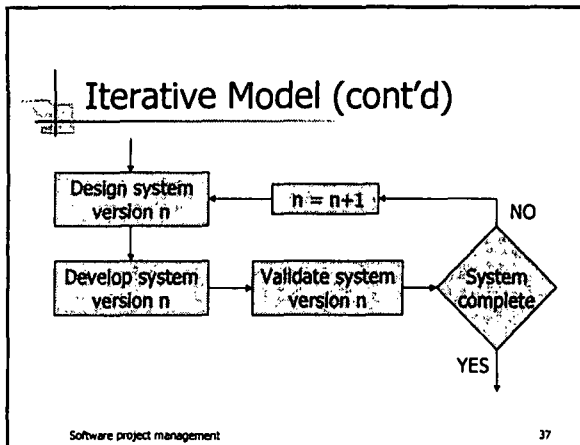
Software project management 34



Iterative Model

- Deliver full system in the beginning
- Enhance functionality in new releases

Software project management 36



- ### Combined Incremental and Iterative Model
- Every new release includes
 - extra functionality
 - enhancement of existing functionality
 - Popularly used in software industry
- Software project management 38

- ### Ranking the Increments
- Rank by value to cost ratio
 - V = value to customer (score 1 -10)
 - C = cost (score 1- 10)
 - Value to cost ratio = V/C
 - Example: Table 4.1
- Software project management 39

- ### Advantages of Phased Development
- Early feedbacks
 - Less possible requirement changes
 - Early benefits for users
 - Improve cash flow
 - Easier control and manage
- Software project management 40

- ### Advantages of Phased Development (cont'd)
- Capture early market
 - Facilitate early training
 - Can be temporarily abandoned
 - Increase job satisfaction
- Software project management 41

- ### Disadvantages of Phased Development
- 'Software breakage'
 - Reduced productivity
- Software project management 42

Operational Specification Model

- Use executables to demonstrate system behaviour
- Resolve requirement uncertainties in early stage
- Merging functionality and system design

Software project management

43

Transformational Model

- Transform a specification into delivered system
- Require automated support
- Rely on formal specification method

Software project management

44

References

- Boehm, B. (1987) A spiral model of software development and enhancement, *Software Engineering Project Management*, p.128–142.
- Hughes, B. and Cotterell, M. (1999) *Software Project Management*, 2nd ed., McGraw Hill.
- Humphrey, W. (1990) *Managing the Software Process*, Addison-Wesley.
- Pfleeger, S.L. (1998) *Software Engineering Theory and Practice*, Prentice Hall

Software project management

45

Software Project Management

Lecture 3 Size and Cost Estimation

Software Project Management

Overview

- Different level of estimation
- Project Evaluation
- Introduction to Estimation
- Size Estimation
- Cost Estimation

Software Project Management

2

Different level of estimation

- Before a project is decided to pursue
 - The estimation is coarse
 - The estimation is in high level terms
 - Profit? Good to the organization? etc.
- After the project is decided to go ahead
 - More detailed size and cost estimations are required

Software Project Management

3

Project Evaluation

- It is a high level assessment of the project
 - to see whether the project is worthwhile to proceed
 - to see whether the project will fit in the strategic planning of the whole organization

Software Project Management

4

Project Evaluation - Why

- Want to decide whether a project can proceed before it is too late
- Want to decide which of the several alternative projects has a better success rate, a higher turnover, a higher ...
- Is it desirable to carry out the development and operation of the software system

Software Project Management

5

Project Evaluation - Who

- Senior management
- Project manager/coordinator
- Team leader

Software Project Management

6

Project Evaluation - When

- Usually at the beginning of the project
 - e.g. Step 0 of Step Wise Framework

Project Evaluation - What

- Strategic assessment
- Technical assessment
- Economical assessment

Project Evaluation - How

- Cost-benefit analysis
- Cash flow forecasting
- Cost-benefit evaluation techniques
- Risk Analysis

Strategic Assessment

- Use to assess whether a project fits in the *long-term goal* of the organization
- Usually carry out by senior management
- Need a strategic plan that clearly defines the objectives of the organization
- Evaluate individual projects against the strategic plan or the overall business objectives

Strategic Assessment (cont'd)

- Programme management
 - suitable for projects developed for use in the organizations
- Portfolio management
 - suitable for project developed for other companies by software houses

SA – Programme Management

- Individual projects as components of a programme within the organization

Programme as "a group of projects that are managed in a coordinated way to gain benefits that would not be possible were the projects to be managed independently"

by D.C. Ferns
Journal of Project Management
Aug. 1991

SA – Programme Management Issues

- Objectives
 - How does the project contribute to the *long-term goal* of the organization?
 - Will the product increase the market share? By how much?

Software Project Management

13

SA – Programme Management Issues (cont'd)

- IS plan
 - Does the product fit into the overall IS plan?
 - How does the product relate to other existing system?

Software Project Management

14

SA – Programme Management Issues (cont'd)

- Organization structure
 - How does the product affect the existing organizational structure? the existing workflow? the overall business model?

Software Project Management

15

SA – Programme Management Issues (cont'd)

- MIS
 - What information does the product provide?
 - To whom is the information provided?
 - How does the product relate to other existing MISs?

Software Project Management

16

SA – Programme Management Issues (cont'd)

- Personnel
 - What are the staff implications?
 - What are the impacts on the overall policy on staff development?
- Image
 - How does the product affect the image of the organization?

Software Project Management

17

SA – Portfolio Management

- suitable for product developed by a software company for an organization
 - outsourcing
- need to assess the product for the client organization
 - Programme management issues apply
- need to carry out strategic assessment for the servicing software company

Software Project Management

18

SA – Portfolio Management Issues

- *Long-term goal* of the software company
- The effects of the project on the portfolio of the company
- Any added-value to the overall portfolio of the company

Technical Assessment

- Functionality against hardware and software
- The strategic IS plan of the organization
- any constraints imposed by the IS plan

Economic Assessment

Why?

- Consider whether the project is the best among other options
- Prioritise the projects so that the resources can be allocated effectively if several projects are underway

Economic Assessment (cont'd)

How?

- Cost-benefit analysis
- Cash flow forecasting
- Various cost-benefit evaluation techniques
 - NPV and IRR

EA – Cost-benefit Analysis

- A standard way to assess the economic benefits
- Two steps
 - Identify and estimate all the costs and benefits of carrying out the project
 - Express the costs and benefits in a common unit for easy comparison (e.g. \$)

EA – Cost-benefit Analysis (cont'd)

- Costs
 - Development costs
 - Setup costs
 - Operational costs

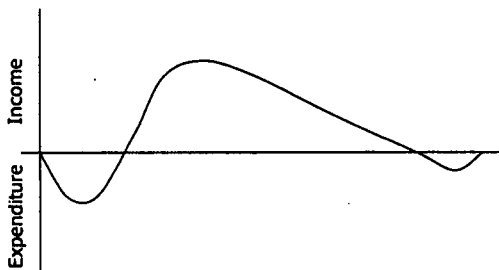
EA – Cost-benefit Analysis (cont'd)

- Benefit
 - Direct benefits
 - Assessable indirect benefits
 - Intangible benefits

EA – Cash Flow Forecasting

- What?
 - Estimation of the cash flow over time
- Why?
 - An excess of estimated benefits over the estimated costs is not sufficient
 - Need detailed estimation of benefits and costs versus time

EA – Cash Flow Forecasting (Cont'd)



EA – Cash Flow Forecasting (Cont'd)

- Need to forecast the expenditure and the income
- Accurate forecast is not easy
- Need to revise the forecast from time to time

Cost-benefit Evaluation Techniques Example

| Year | Project 1 | Project 2 | Project 3 | Project 4 |
|------------|-----------|------------|-----------|-----------|
| 0 | -100,000 | -1,000,000 | -100,000 | -120,000 |
| 1 | 10,000 | 200,000 | 30,000 | 30,000 |
| 2 | 10,000 | 200,000 | 30,000 | 30,000 |
| 3 | 20,000 | 200,000 | 30,000 | 30,000 |
| 4 | 20,000 | 200,000 | 20,000 | 25,000 |
| 5 | 100,000 | 350,000 | 20,000 | 50,000 |
| Net Profit | 60,000 | 150,000 | 30,000 | 45,000 |
| Payback | 5 | 5 | 4 | 4 |
| ROI | 12% | 4% | 10% | 11% |

Cost-benefit Evaluation Techniques

- Net profit
 - = Total income – Total costs
- Payback period
 - = Time taken to break even
- Return on Investment (ROI)
 - = $\frac{\text{average annual profit}}{\text{total investment}} \times 100\%$

Cost-benefit Evaluation Techniques – NPV

Net present value (NPV)

- It is the sum of the present values of all future amounts.
- *Present value* is the value of which a future amount worth at present
- It takes into account the profitability of a project and the timing of the cash flows

Cost-benefit Evaluation Techniques – NPV (cont'd)

- *Discount rate* is the annual rate by which we discount future earning
 - e.g. If discount rate is 10% and the return of an investment in a year is \$110, the present value of the investment is \$100.

Cost-benefit Evaluation Techniques – NPV (cont'd)

- Let n be the number of year and r be the discount rate, the present value (PV) is given by

$$PV = \frac{\text{value in year } n}{(1+r)^n}$$

Cost-benefit Evaluation Techniques – NPV (cont'd)

- Issues in NPV
 - Choosing an appropriate discount rate is difficult
 - Ensuring that the rankings of projects are not sensitive to small changes in discount rate

Cost-benefit Evaluation Techniques – NPV (cont'd)

- Guidelines:
 - Use the standard rate prescribed by the organization
 - Use interest rate + premium rate
 - Use a target rate of return
 - Rank the projects using various discount rates

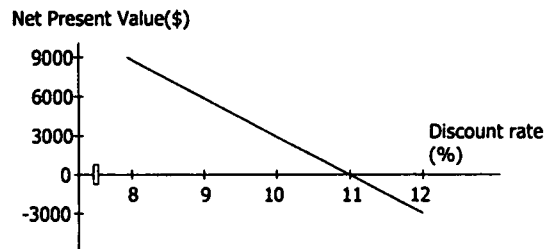
Cost-benefit Evaluation Techniques – NPV (cont'd)

- Disadvantage
 - May not be directly comparable with earnings from other investments or the costs of borrowing capital

Cost-benefit Evaluation Techniques – IRR

- Internal Rate of Return (IRR)
 - The percentage discount rate that would produce a NPV of zero
 - A relative measure

Cost-benefit Evaluation Techniques – IRR (cont'd)



Cost-benefit Evaluation Techniques – IRR (cont'd)

- Advantages
 - Convenient
 - Directly comparable with rate of return on other projects and with interest rates
 - Useful
 - Dismiss a project due to its small IRR value
 - Indicate further precise evaluation of a project
 - Supported by MS Excel and Lotus 1-2-3

Estimation

- Why? – to define the project budget and to 'refine' the product to realize the budget
- Who? – the manager
- What? – size and cost
- When? – always
- How? – techniques and models

Issues related to Estimation

- Difficult to make accurate estimation
- Better to have previous data and analyze the actual values against their estimates so that you know how accurate you are
- Even better to have previous data of the whole organization so that you know how accurate the estimation method, if any, used within the organization is

Positive Attitude Towards Estimation

- Use your estimation as a guide to manage your project
- From time to time, you need to revise your estimation based on the current status of the project

Estimation Approaches

- Expert judgement
 - Ask the knowledgeable experts
- Estimation by analogy
 - Use the data of a similar and completed project
- Pricing to win
 - Use the price that is low enough to win the contract

Estimation Approaches (cont'd)

- Top-down
 - An overall estimate is determined and then broken down into each component task
- Bottom-up
 - The estimates of each component task are aggregate to form the overall estimate
- Algorithmic model
 - Estimation is based on the characteristics of the product and the development environment.

Size Estimation

- Problems related to size estimation
- Size Estimation Model
 - Function Point Analysis (FPA)

Problems related to size estimation

- Nature of software
- Novel application of software
- Fast changing technology
- Lack of homogeneity of project experience
- Subjective nature of estimation
- Political implication with the organization

Function Point Analysis (FPA)

- Developed by A. Albrecht in IBM
- Aim: To estimate the LOC of a system

LOC of system

$$= \text{FP of system} \times \text{LOC-per-FP of the language}$$

Function Point Analysis (cont'd)

- Idea: Software system comprises of five major components (or, *external user type*)
 - External input types
 - External output types
 - Logical internal file types
 - External interface file types
 - External inquiry types

Function Point Analysis - Steps

- Identify each instance of each external user type in the proposed system
- Classify each instance as having high, medium or low complexity
- Assign the FP of each instance
- FP of the system = sum of FP of individual components

Function Point Analysis

| <i>Number of FPs</i> | <i>Complexity</i> | | |
|------------------------------|-------------------|---------|------|
| | Low | Average | High |
| External user type | | | |
| External input type | 3 | 4 | 6 |
| External output type | 4 | 5 | 7 |
| Logical internal file type | 7 | 10 | 15 |
| External interface file type | 5 | 7 | 10 |
| External inquiry type | 3 | 4 | 6 |

Function Point Analysis - Example

- A component of an inventory system consisting of 'Add a record', 'Delete a record', 'Display a record', 'Edit a record', and 'Print a record' will have
 - 3 external input types
 - 1 external output type
 - 1 external inquiry type
- Then, assign FPs based on the complexity of each external types

Function Point Analysis (cont'd)

- Other issues
 - The assignment of level of complexity is rather subjective
 - International FP User Group (IFPUG) imposes rules on assigning the level of complexity to individual external user types

Object Point Analysis

- Similar to function point analysis
- Used on 4GL development projects
- Take account of features that may be more readily identifiable if the system is built on a high-level application building tools

Object Point Analysis – Steps

- Identify the number of screens, reports and 3GL components
 - Classify each object as Simple, Medium and Difficult
 - Assign the weight accordingly
 - Calculate the total object points
- Total OP = sum of individual OP × weighting

Object Point Analysis – Steps (cont'd)

- Deduct the reused objects (r% reused)

$$\text{NOP} = \text{OP} \times (1 - r\%)$$
- Identify the productivity rate of both developer and CASE
- Productivity rate = average of the two PRs
- Calculate the effort

$$\text{Effort} = \text{NOP} / \text{Productivity Rate}$$

Software Project Management

55

Object Point Analysis – Screens

| Number of views contained | Number and source of data tables | | |
|---------------------------|-------------------------------------|---------------------------------------|------------------------------------|
| | Total < 4 (<2 server, <2 client) | Total < 8 (2-3 server, 3-5 client) | Total 8+ (>3 server, >5 client) |
| < 3 | Simple | Simple | Medium |
| 3 – 7 | Simple | Medium | Difficult |
| 8+ | Medium | Difficult | Difficult |

Software Project Management

56

Object Point Analysis – Reports

| Number of sections contained | Number and source of data tables | | |
|------------------------------|-------------------------------------|---------------------------------------|------------------------------------|
| | Total < 4 (<2 server, <2 client) | Total < 8 (2-3 server, 3-5 client) | Total 8+ (>3 server, >5 client) |
| < 2 | Simple | Simple | Medium |
| 2 or 3 | Simple | Medium | Difficult |
| > 3 | Medium | Difficult | Difficult |

Software Project Management

57

Object Point Analysis – Complexity Weightings

| Type of object | Complexity | | |
|----------------|------------|--------|-----------|
| | Simple | Medium | Difficult |
| Screen | 1 | 2 | 3 |
| Report | 2 | 5 | 8 |
| 3GL component | N/A | N/A | 10 |

Software Project Management

58

Object Point Analysis – Productivity Rate

| | Very low | Low | Nominal | High | Very High |
|---------------------------------------|----------|-----|---------|------|-----------|
| Developer's experience and capability | 4 | 7 | 13 | 25 | 50 |
| CASE maturity and capability | 4 | 7 | 13 | 25 | 50 |

Software Project Management

59

Object Point Analysis – Issues

- Adopted in Boehm's COCOMO II in the application composition stage

Software Project Management

60

Object Point Analysis – Example

- See separate handout

Cost Estimation

- Cost Estimation Model
 - COCOMO II

Constructive Cost Model II (COCOMO II)

- A parametric cost model
 - Important aspects of software project are characterized by variables (or, parameters)
 - Once the value of the parameters are determine, the cost can be computed from the equation

COCOMO II (cont'd)

- Recognize different approaches to software development
 - Prototyping, Incremental development etc.

A history of COCOMOs

- *COCOMO* originally proposed by Boehm in 1981, now called *COCOMO 81*
- Later evolved to *Ada COCOMO* in 1989
- In 1995, Boehm proposes *COCOMO II*

COCOMO II

- A family of models
 - Use different models in 3 different stages of the project
- 3 stages: application composition, early design and post architecture
 - Support estimation early in the process
 - Allow further detailed estimation after the system architecture has been defined

COCOMO II (cont'd)

- The basic model equation

$$\text{Effort} = \text{Constant} \times (\text{Size})^{\text{scale factor}} \times \text{Effort Multiplier}$$
 - Effort in terms of person-month
 - Size: Estimated Size in KSLOC
 - Scale Factor: a combined effects of factors related to the process
 - Effort Multiplier (EM): a combined effect of factors related to the effort

The Application Composition Stage

- Estimation at the early stage
- Corresponding to exploratory work such as prototyping
- Use object points to estimate the size of the product

The Early Design Stage

- Estimate after the requirements specification is completed and possibly with some design
- Use the basic model equation
- Estimate the size by FPs (preferred) or KSLOC
- Assign process exponent estimation accordingly

The Early Design Stage – Scale Factor

- Estimation on the scale factor
 - A combined effect of 5 parameters
- Application precedentedness
- Process flexibility
- Architecture risk resolution
- Team cohesion
- Process maturity

The Early Design Stage – Scale Factor (cont'd)

| Parameter | Very Low (0.05) | Low (0.04) | Nominal (0.03) | High (0.02) | Very High (0.01) | Extra High (0.00) |
|------------------------------|-----------------------------|-----------------------------|------------------------|---------------------|--------------------|-----------------------|
| Precedentedness | Thoroughly unprecedented | Largely unprecedented | Somewhat unprecedented | Generally familiar | Largely familiar | Thoroughly familiar |
| Development flexibility | Rigorous | Occasional relaxation | Some relaxation | General conformity | Some conformity | General goals |
| Architecture risk resolution | Little 20% | Some 40% | Often 60% | Generally 75% | Mostly 90% | Full 100% |
| Team cohesion | Very difficult interactions | Some difficult interactions | Basically cooperative | Largely cooperative | Highly Cooperative | Seamless interactions |
| Process maturity | Level 1 | Level 2 | Level 2+ | Level 3 | Level 4 | Level 5 |

The Early Design Stage – Scale Factor (Cont'd)

- Calculate the scale factor based on the equation

$$\text{Scale factor} = 1.01 + \text{sum of the values}$$

The Early Design Stage – Effort Multiplier

- 7 factors on Effort Multiplier
 - product Reliability and ComPleXity (RCPX)
 - required reusability (RUSE)
 - Platform DIfficulty (PDIF)
 - PeRsonnel capability (PERS)
 - PeRsonnel EXperience (PREX)
 - FaCILities available (FCIL)
 - SCHEDule pressure (SCED)

Software Project Management

73

The Early Design Stage – Effort Multiplier (cont'd)

- Assess each factor by
 - Very low, low, nominal, high, very high, and extra high
- Assign each factor using a value between 0.5 and 1.5 (inclusive)
- EM is the product of all these values

Software Project Management

74

The Early Design Stage – Effort Multiplier (cont'd)

| Early Design | Very Low – Extra High |
|--------------|-----------------------|
| RCPX | 0.5 – 1.5 |
| RUSE | 0.5 – 1.5 |
| PDIF | 0.5 – 1.5 |
| PERS | 1.5 – 0.5 |
| PREX | 1.5 – 0.5 |
| FCIL | 1.5 – 0.5 |
| SCED | 1.5 – 0.5 |

Software Project Management

75

The Early Design Stage – Example

- See separate handout

Software Project Management

76

The Post-architecture Stage

- Estimation after the software architecture has been defined
- The same basic model equation
- Size estimation by KSLOC (preferred) or FPs
- Same process exponent estimation
- 17 factors in EM (more than 7 in early design stage)

Software Project Management

77

The Post-architecture Stage – Effort Multiplier

- 17 factors in 4 different categories
 - Product attributes
 - Platform attributes
 - Personnel attributes
 - Project attributes

Software Project Management

78

The Post-architecture Stage – Effort Multiplier

- Product attributes
 - Required reliability (RELY)*
 - Database size (DATA)
 - Product complexity (CPLX)*
 - Required reuse (RUSE)**
 - Documentation (DOCU)
- *Relate to RCPX in early design stage

The Post-architecture Stage – EAF (Cont'd)

- Platform attributes
 - execution TIME constraint (TIME)*
 - main STORAge constraint (STOR)*
 - Platform VOLatility (PVOL)*
- *Related to Platform DIFFiculty (PDIF) in early design stage

The Post-architecture Stage – EAF (Cont'd)

- Personnel attributes
 - Analyst CAPabilities (ACAP)^
 - Application EXPerience (AEXP)*
 - Programmer CAPabilities (PCAP)^
 - Personnel EXPerience (PEXP)*
 - programming Language/Tool EXperience (LTEX)*
 - Personnel CONTinuity (PCON)^

The Post-architecture Stage – EAF (Cont'd)

- Project attributes
 - use of software TOOLS (TOOL)*
 - multiSITE development team communications (SITE)*
- *Relate to FCIL in early design model

EAF Relations

| Early Design | Post-Architecture |
|--------------|------------------------|
| RCPX | RELY, DATA, CPLX, DOCU |
| RUSE | RUSE |
| PDIF | TIME, STOR, PVOL |
| PERS | ACAP, PCAP, PCON |
| PREX | AEXP, PEXP, LTEX |
| FCIL | TOOL, SITE |
| SCED | SCED |

The Post-architecture Stage – Example

- See separate handout

COCOMO II (cont'd)

- Advantages
 - Good improvement over COCOMO
 - Good match for iterative development, modern technology, and management process
- Disadvantages
 - Still immature, diverse projects in database
 - Hard to believe that it will be any more reliable than the original COCOMO model

References

- Hughes, B., and Cotterell, M. (1999) *Software project management*, 2nd ed., McGraw Hill
- Pfleeger, S.L. (1998) *Software Engineering: Theory and Practice*, Prentice Hall
- Royce, W. (1998) *Software Project Management: A Unified Framework*, Addison Wesley
- Center for Software Engineering, USC (1999) *COCOMO II Model Definition Manual*.

Software Project Management

Lecture 4 Software Project Planning

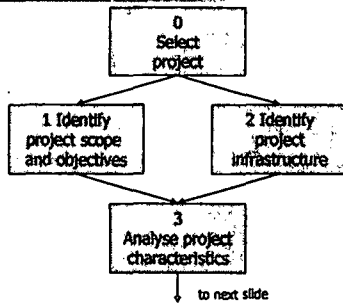
Overview

- Step Wise project planning framework
- Preparation of a software project plan
- Planning and scheduling the activities in software project management
- Various approaches towards activity plan
- Various scheduling techniques such as sequencing and CPM

Software Project Management

2

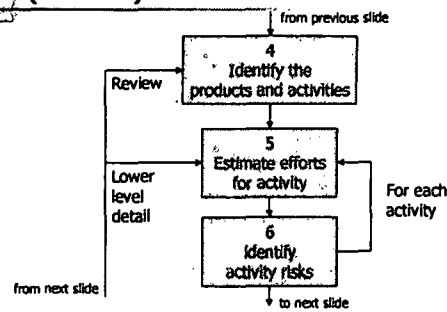
Step Wise – An Overview



Software Project Management

3

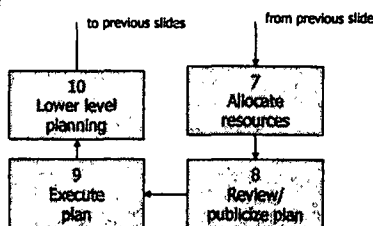
Step Wise – An Overview (cont'd)



Software Project Management

4

Step Wise – An Overview (cont'd)



Software Project Management

5

Step Wise – An Overview (cont'd)

- Step 0: Select project
- Step 1: Identify project scope and objectives
- Step 2: Identify project infrastructure
- Step 3: Analyze project characteristics
- Step 4: Identify project products and activities

Software Project Management

6

Step Wise - An Overview (cont'd)

- Step 5: Estimate effort for each activity
- Step 6: Identify activity risks
- Step 7: Allocate resources
- Step 8: Review/publicize plan
- Step 9: Execute plan
- Step 10: Execute lower levels of planning

Software Project Management

7

Step 1: Identify Project Scope and Objectives

- Step 1.1 Identify objectives and practical measures of the effectiveness in meeting those objectives
- Step 1.2 Establish a project authority
 - To ensure the unity of purpose among all persons concerned

Software Project Management

8

Step 1: Identify Project Scope Objectives (cont'd)

- Step 1.3 Identify all stakeholders in the project and their interests
- Step 1.4 Modify objectives in the light of stakeholder analysis
- Step 1.5 Establish methods of communication between all parties

Software Project Management

9

Step 2: Identify Project Infrastructure

- Step 2.1 Identify relationship between the project and strategic planning
 - To determine the order of related projects (in the organization) being carried out
 - To establish a framework within which the system fits
 - To ensure the hardware and software standards are followed

Software Project Management

10

Step 2: Identify Project Infrastructure (cont'd)

- Step 2.2 Identify installation standards and procedures
 - more appropriate name: "Identify standards and procedures related to the software project"
- Step 2.3 Identify project team organization

Software Project Management

11

Step 3: Analyse Project Characteristics

- Step 3.1 Distinguish the project as either objective-driven or product-driven
- Step 3.2 Analyse other project characteristics (including quality-based ones)
- Step 3.3 Identify high level project risks
- Step 3.4 Take into account user requirements concerning implementation

Software Project Management

12

Step 3: Analyse Project Characteristics (cont'd)

- Step 3.5 Select general lifecycle approach in the light of the above
 - Step 3.6 Review overall resource estimates
 - Up to this stage,
 - the major risks of the project are identified
 - the overall approach of the project is decided
- So, it is a good place to re-estimate the required effort and other resources for the project

Software Project Management

13

Step 4: Identify Project Products and Activities

- Step 4.1 Identify and describe project products
 - Identify all the products related to the project
 - Account for the required activities
- Step 4.2 Document generic product flows
 - To document the relative order of the products
- Step 4.3 Recognize product instances

Software Project Management

14

Step 4: Identify Project Products and Activities(cont'd)

- Step 4.4 Produce an ideal activity network
 - Activity network shows the tasks that have to be carried out as well as their sequence of execution for the creation of a product from another
- Step 4.5 Modify the ideal to take into account need for stages and checkpoints
 - To check compatibility of products of previous activities

Software Project Management

15

Step 5: Estimate Effort for Each Activity

- Step 5.1 Carry out bottom-up estimates
 - need to estimate staff effort, time for each activity and other resources
- Step 5.2 Revise plan to create controllable activities
 - need to break a task into a series of manageable sub-tasks

Software Project Management

16

Step 6: Identify Activity Risks

- Step 6.1 Identify and quantify the risks of each activity
- Step 6.2 Plan risk reduction and contingency measures where appropriate
- Step 6.3 Adjust overall plans and estimates to take account of risks

Software Project Management

17

Step 7: Allocate Resources (Staffing)

- Step 7.1 Identify and allocate resources
 - type of staff needed for each activity
 - staff availability are identified
 - staff are provisionally allocated to task
- Step 7.2 Revise plans and estimates to take into account resource constraints
 - staffing constraints
 - staffing issues

Software Project Management

18

Step 8: Review/publicize Plan

- Step 8.1 Review quality aspects of the project plan
 - To ensure each activity is completed with a quality product
 - Each activity should have 'exit requirements'.
 - This ensures the quality of the product on each activity.

Step 8: Review/publicize Plan (cont'd)

- Step 8.2 Document plans and obtain agreement
 - all parties understand and agree to the commitments in the plan

Aside – When to plan

- Planning is an on-going process of refinement
- Planning at different stages of the project have different emphases and purposes

Project Vs Activity

- A project is composed of a number of related activities
- A project may start when at least one of its activities is ready to start
- A project will be completed when all of its activities have been completed

Project Vs Activity (cont'd)

- An activity must have a clear start and a clear stop
- An activity should have a duration that can be forecasted
- Some activities may require that other activities are completed before they can begin

Activity Planning

- A project plan is a schedule of activities indicating the start and stop for each activity
 - Also provide the project and resource schedules
- The start and stop of each activity should be visible and easy to measure
- Each activity should have some 'deliverables' for ease of monitoring

Activity Planning (cont'd)

- During planning, managers consider:
 - Resource availability
 - Resource allocation
 - Staff responsibility
 - Project Monitoring
 - Cash flow forecasting
 - Re-planning of the project towards the pre-defined goal

Other Objectives of Activity Planning

- Feasibility assessment
- Resources allocation
- Detailed costing
- Motivation
- Co-ordination

Different Levels of Plans

- Project Schedule: a plan that shows
 - 1. the dates when each activity should start and stop
 - 2. when and how much of the resources will be required
- Activity Plan: a plan that describe
 - how each activity would be undertaken

Project Schedule in 4 Stages

- Ideal Activity Plan
 - An activity plan without any constraints
- Risk consideration for each activity
- Resource consideration for whole project
- Schedule production (known to 'public')

Various Approaches Towards Identifying Activity

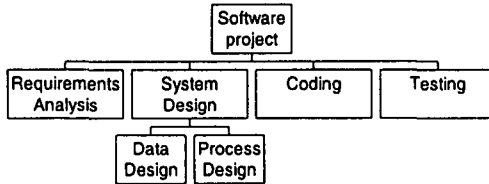
- Activity-based approach
- Product-based approach
- Hybrid approach

Activity-based Approach

- Use *Work Breakdown Structure (WBS)* to generate a task list
- WBS involves
 - identifying the main tasks
 - break each main task down into subtasks
 - The subtasks can further be broken down into lower level tasks.

Activity-based Approach (cont'd)

Work Breakdown Structure (an extract)



Activity-based Approach (cont'd)

- Advantages
 - More likely to obtain a task catalogue that is complete and is composed of non-overlapping tasks
 - WBS represents a structure that can be refined as the project proceeds
 - The structure already suggests the dependencies among the activities

Activity-based Approach (cont'd)

- Disadvantage
 - Very likely to miss some activities if an unstructured activity list is used

Product-based Approach

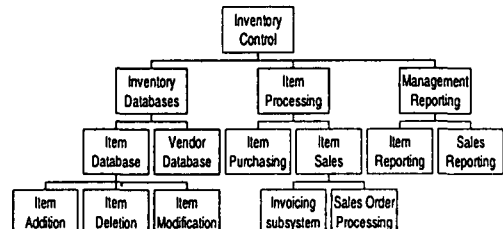
- Product Breakdown Structure (PBS)
 - To show how a system can be broken down into different products for development
- Product Flow Diagram (PFD)
 - To indicate, for each product, which products are required as 'inputs'

Product-based Approach (cont'd)

- Advantages
 - Less likely to miss a product unexpectedly from a PBS

Product-based Approach – An example

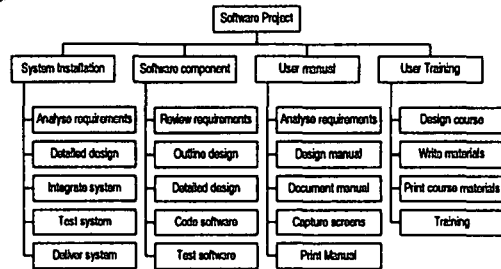
A Product Breakdown Structure (an extract)



Hybrid Approach

- A mix of the activity-based approach and the product-based approach
- More commonly used approach
- The WBS consists of
 - a list of the products of the project; and
 - a list of activities for each product

Hybrid Approach (cont'd)



Hybrid Approach (cont'd)

- IBM in its MITP methodology suggests 5 levels
 - Level 1: Project
 - Level 2: Deliverables (software, manuals etc)
 - Level 3: Components
 - Level 4: Work-packages
 - Level 5: Tasks (individual responsibility)

Planning and Scheduling the Activities

- Once we have a project plan (or, project schedule), we need to schedule the activities in a project taking into the resource constraint

Scheduling Techniques

- Simple sequencing
 - Suitable for small projects
- Critical Path Method (CPM)
 - Suitable for large software projects
 - The most commonly used "networking" techniques

Simple sequencing

- A simple sequencing of the tasks and the responsible personnel taken into account of the resources
- Easily presented in a simple bar chart
 - see figure 6.6 in Hughes book
- Suitable to allocate individuals to particular tasks in early stage

Critical Path Method (CPM)

- Primary objectives:
 - Planning the project so that it can be completed as quickly as possible
 - Identifying those activities where their delays is likely to affect the overall project completion date
- Developed by Du Pont Chemical Company and published in 1958

Critical Path Method (cont'd)

- Capture the activities and their inter-relationships using a graph
 - Lines are used to represent the activities
 - Nodes are used to represent the start and stop of activities

Critical Path Method (cont'd)

- Adding time dimension
 - The forward pass
 - calculate the earliest start dates of the activities
 - To calculate the project completion date
 - The backward pass
 - calculate the latest start dates for activities
 - help to identify the critical path from the graph

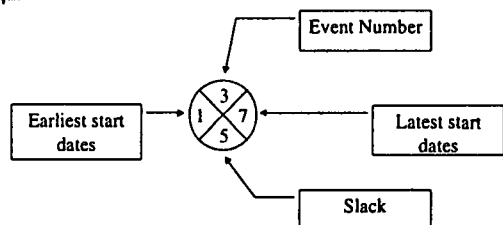
Critical Path Method (cont'd)

- Identifying critical path and critical event
 - Critical event: an event that have zero *slack*
 - Critical path: a path joining those critical events

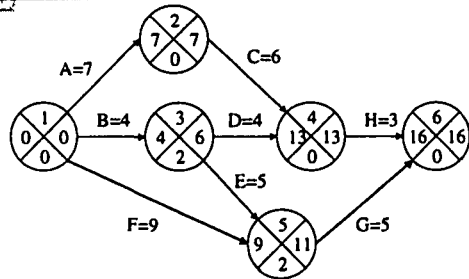
Example to construct a CPM

| Id. | Activity Name | Duration (weeks) | Precedents |
|-----|-----------------------|------------------|------------|
| A | Hardware selection | 7 | |
| B | Software design | 4 | |
| C | Hardware Installation | 6 | A |
| D | Coding | 4 | B |
| E | Data Preparation | 5 | B |
| F | User Documentation | 9 | |
| G | User Training | 5 | E,F |
| H | System Installation | 3 | C,D |

Example to construct a CPM (cont'd)



Example to construct a CPM (cont'd)



Software Project Management

49

Activity Float

- Time allowed for an activity to delay
- 3 different types:
 - Total float (without affecting the completion of the project)
 - = latest start date - earliest start date
 - Free float (without affecting the next activity)
 - = earliest start date of *next* activity - latest end date of *previous* activity
 - Interfering float (= total float - free float)

Software Project Management

50

Significance of critical path

- During planning stage,
 - Shortening the critical path will reduce the overall project duration
- During management stage,
 - Pay more attention to those activities which fall in the critical path

Software Project Management

51

References

- Hughes, B., and Cotterell, M. (1999) *Software Project Management*, 2nd edition, McGraw-Hill.
- Pfleeger, S.L. (1998) *Software Engineering: Theory and Practice*, Prentice Hall.

Software Project Management

52

Software Project Management

Lecture 5 Software Project Risk Management

Overview

- Project risks
- Nature of risks
- Risk identification
- Risk estimation
- Risk evaluation
- Risk management
- Risk reduction strategies

Software Project Management

2

Project Risks

- Factors that cause a project to delay or over-budget

Software Project Management

3

Nature of Project Risks

- Planning assumptions
- Estimation errors
- Eventualities

Software Project Management

4

Planning Assumptions

- Why assumptions
 - Uncertainties in early stage of the project

Software Project Management

5

Planning Assumptions (cont'd)

- Common assumption:
 - "Everything goes smooth" symptom
 - Software is OK
 - Design is perfect in the first place
 - Coding is 'nearly perfect'

Software Project Management

6

Planning Assumptions (cont'd)

- 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

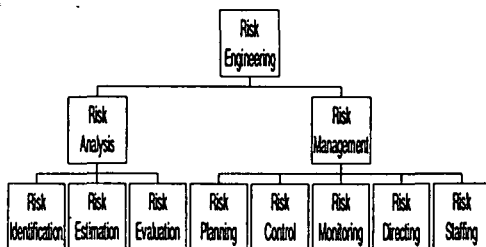
Estimation Error (cont'd)

- 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
 - Requirement specification needs to be rewritten
 - Staffing problem

Boehm's Risk Engineering



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

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 apply 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 problem

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?

Software Project Management

19

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

Software Project Management

20

Supplier Factors

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

Software Project Management

21

Environment Factors

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

Software Project Management

22

Health and Safety Factors

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

Software Project Management

23

Boehm's Top Ten Risk Items

- Personnel shortfalls
- Unrealistic schedules and budgets
- Developing the wrong software functions
- Developing the wrong user interface
- Gold Plating

Software Project Management

24

Boehm's Top Ten Risk Items (cont'd)

- Continuing stream of requirements changes
- Shortfalls in externally performed tasks
- Shortfalls in externally furnished components
- Real-time performance shortfalls
- Straining computer science capabilities

Risk Estimation

- Recall that
Hazard → Problem → Risk
- Risk estimation is to assess the likelihood and impact of each hazard
- Risk exposure (risk value)
 - It is the importance of the risk
$$\text{Risk exposure} = \text{risk likelihood} \times \text{risk impact}$$

Risk Estimation (cont'd)

- Risk likelihood
 - The probability that a hazard is going to occur
- Risk impact
 - The effect of the problem caused by the hazard

Risk Estimation (cont'd)

- Advantages
 - The only way to compare or rank the risks
 - To have a good quantitative estimate, the extra effort can provide a better understanding of the problem
- Disadvantages
 - Estimation is difficult, subjective, time-consuming and costly

Risk Estimation Techniques

- Risk likelihood
 - Rank from Low, Medium to High
 - Rank from 1 (least likely) to 10 (most likely)
- Risk Impact
 - Rank from 1 to 10

Risk Evaluation

- Ranking the risks
- Determining the corresponding risk aversion strategies



Ranking Risks

- Ranking the risks based on their risk exposures
- Ranking shows the order of importance
- In practice, also consider factors like
 - Confidence of the risk assessment
 - Compound risk
 - The number of risks
 - Cost of action



Risk Reduction Leverage (RRL)

- RRL is used to determine whether it is worthwhile to carry out the risk reduction plan.
- The higher is the RRL value, the more worthwhile is to carry out the risk reduction plan.

$$RRL = \frac{RE_{before} - RE_{after}}{\text{risk reduction cost}}$$



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
- Distinctions among them are fuzzy

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 programmer

References

- Boehm, B. (1989) *Tutorial on Software Risk Management*, IEEE CS Press
- Hughes, B., and Cotterell, M. (1999) *Software Project Management*, 2nd edition, McGraw-Hill
- Pfleeger, S.L. (1998) *Software Engineering: Theory and Practice*, Prentice Hall

Software Project Management

Lecture 6 Resource Allocation

Lecture Overview

- Resource Identification
- Resource Distribution
- Resource Scheduling

Software Project Management 2

What is Resource Allocation

- After the activities have been identified using various techniques and tabulated into a Work-Break-Down the resource need to be allocated to complete the identified tasks. This process is considered resource allocation.

Software Project Management 3

Who allocates resources?

- Project Manager.
 - Concentrate on resource where there is a possibility that, without planning they might not be sufficiently available when required.
 - Senior Software Developers are the hardest to find – these need to be very carefully planned for in advance.
 - Developers do not like to wait for work, they prefer to be busy with activities and tasks that show clear progress.

Software Project Management 4

Result of Resource Allocation

- Reflected in many schedules,
 - Activity Schedule.
 - Resource Schedule.
 - Cost Schedule.
- Changes to these schedules are very much interrelated and require domain experience to “get it right”.

Software Project Management 5

Resource Categories

- Labour (Even the project manager).
- Equipment (Coffee Machine?).
- Materials (Consumed items – floppy disks).
- Space (Rooms, Cubicles).
- Services (Telecomm, Cleaning services).
- Time (The most rigid item of all).
- Money (Secondary resource).

Note: These are broad categories only.

Software Project Management 6

Resource Organisation

- A program organization chart is essential to allocate staff effectively,
 - Develop the hierarchical program organization.
 - Identify Roles and Responsibilities.
 - Plan for number of staff in each role (at a high level).
 - Establish Teams.

Software Project Management

7

Resource Requirement Identification – 1

- For each activity identify,
 - Work amount required (in work units)
 - Basic Skill level required (to even undertake the task)
 - Complexity of the task (this will help to determine the experience required)
 - Task Category (Unskilled, skilled, leadership, expert, management)

Software Project Management

8

Resource Requirement Identification – 2

- Example.
 - Activity – Install Network Hardware for 20 computers.
 - Work units - 20.
 - Basic Skill – Bachelors Degree in related field.
 - Task Complexity: 5.
 - Task Category: Skilled (other categories may be Management, Leadership, Expert)

Software Project Management

9

Resource Scheduling

- After all the required resources have been identified, they need to be scheduled effectively.
- The earliest start dates, last start dates will need to be taken into account to schedule resources efficiently.
- Resources should be balanced throughout the project.

Software Project Management

10

Resource Scheduling – 2

- Human resource scheduling issues,
 - Planned Leave, Public Holidays.
 - Possible Sick leave (random, subjective at best and hard to predict).
 - General motivation and enthusiasm for the task allocated (If they dislike the task, it will flow through into the output).
 - Work load and stress in project.
 - Stress outside work.

Software Project Management

11

Resource Histograms

- Commonly used during planning to indicate possible problem areas,
 - People (by category) Vs Week Number
 - For each individual – estimated number of tasks (including complexity) over weeks
 - This helps in reducing work load some times to help the individual recover from any heavy load.
 - Category Vs Week

Software Project Management

12

External Dependencies

- When planning any resources that rely on external factors need to be planned with associated risks involved.

Software Project Management

13

Parallel, Sequential Tasks

- Tasks run both in parallel and sequentially.
- Depending on the activity network and critical path, resource allocation needs to be planned effectively.
- Competing tasks need to be prioritised with risk before resource allocation.

Software Project Management

14

Prioritisation Techniques

- Total Float Priority
- Ordered List Priority

- There are many others that refine on top of these, but broadly these cover the general cases well.

Software Project Management

15

Total Float Priority

- Ordered according to their total float.
- Smallest total float has highest priority.
- Activities are allocated resources in ascending order of total float.
- Changes to plan will require re-calculation.

Software Project Management

16

Ordered List Priority

- Activities that can proceed at the same time are ordered according to a set of simple criteria.
- Burman's priority list takes into account activity duration as well as total float:
 - Shortest critical activity.
 - Critical activities.
 - Shortest non-critical activity.
 - Non-critical activity with least float.
 - Non-critical activities.
- Note: Other ways of ordering is also allowed.

Software Project Management


17

Critical Paths

- Resource scheduling will almost always change the activity network.
- The changes often result in changes to the critical path.
 - Delaying an activity due to lack of correct resources will cause that activity to become critical after it uses up all its slack time.
- These changes are often experienced after the project has started which will require adapting during the project (this is normally much harder in practice).

Software Project Management


18



Cost of Resources

- All projects concentrate on completing it in the shortest time span with minimum resources (in planning stage).
- However, once the project starts – all un-planned for issues and any risks will cause some strain on the cost.


Software Project Management 19



Resource Allocation Issues

- Availability
- Criticality
- Risk
- Training
- Team Building


Software Project Management 20



Cost Scheduling

- Broad Categories,
 - Staff.
 - Overheads (Office Space, Interest charges, Travel Costs, Insurance and so on).
 - Usage charges (for external resources or contractors, leased/rental equipment).


Software Project Management 21



Scheduling in Practice

- It should always be in the project planners mind, right from the start of the project.
- During the resource scheduling and allocation phase of the planning activity – a lot of the plan will change.
- Most of the issues with respect to resource allocation and scheduling arise after the project starts (normally after about 30% of the activities are complete).


Software Project Management 22



Summary

- Identify all resources required.
- Arrange activity start/end dates to minimise variations in resource levels over the duration of the project.
- Allocating resources to competing activities in a rational order of priority.
- Critical/High-Risk activities should be backed up the experienced staff.

Software Project Management 23



References

- Hughes. B and Cotterell. M, *Software Project Management*, 2nd Edition, McGraw Hill 1999.
- Cantor. M. R, *Object Oriented Project Planning with UML*, Wiley 2000.

Software Project Management 24

Software Project Management

Lecture 7
ISO 9000

Overview

- ISO 9000 family of standards
- Overview of ISO 9001
- Three levels of quality assurance
- Manufacturing industry versus software industry
- Twenty quality elements in ISO 9000
- Characteristics of an ISO 9000 quality system

Software Project Management

2

Overview (cont'd)

- Satisfying ISO 9000
- Introduction of ISO 9000-3
- Assumptions of ISO 9000-3
- Overview of ISO 9000-3
- TickIT Initiative
- Why comply with ISO 9001
- Potential problems of ISO 9001

Software Project Management

3

ISO 9000 Family of Standards

- A series of international quality standards developed by the International Organization for Standardization
- Originally developed for two-party contractual situations, mainly for the manufacturing environment

Software Project Management

4

ISO 9000 Family of Standards (cont'd)

- Applies to the quality management system and the process used to produce a product
- Ensures that the process can consistently produce products that meet the expectation of the customers

Software Project Management

5

ISO 9000 Family of Standards (cont'd)

- Provides a framework for improving business processes
- Does NOT provide for leading-edge quality, but does provide a strong quality foundation upon which a company can build

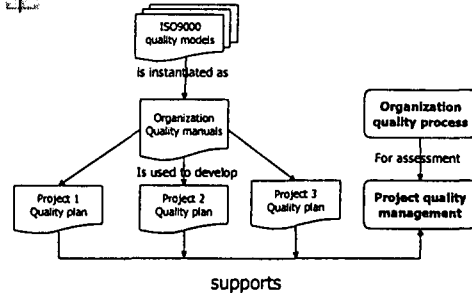
Software Project Management

6

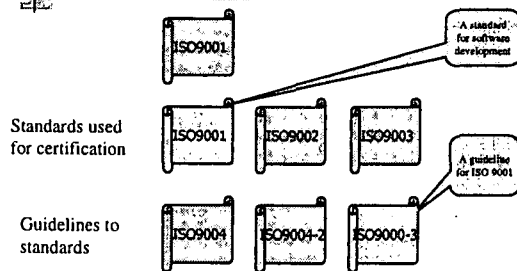
ISO 9000 Family of Standards (cont'd)

- Provide a generic model of the quality process; must be instantiated for each organization
- Describe what, at the minimum, must be done; does NOT specify how things are to be done

ISO 9000 and Quality Management



Guidelines for selection and use of the ISO 9000 standards



ISO 9000 Family of Standards (cont'd)

- ISO 9000-1 is a general guideline which gives background information about the family of standards
- ISO 9001, ISO 9002, and ISO 9003 are standards in the family, containing requirements on a supplier

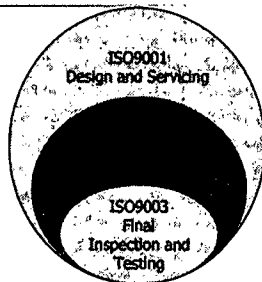
ISO 9000 Family of Standards (cont'd)

- ISO 9002 and ISO 9003 are subsets of ISO 9001
 - ISO 9002 applies when there is no design
 - ISO 9003 applies when there is neither design nor production

ISO 9000 Family of Standards (cont'd)

- ISO 9004 is a comprehensive guideline to the use of the ISO 9000 standards
- For software development, ISO 9001 is the standard to use
- ISO 9000-3 is a guideline on how to use ISO 9001 for software development
- ISO 9004-2 is a guideline for the application of ISO 9001 to the supply of services (including computer centers and other suppliers of data services)

Relationship of ISO 9000 standards



Software Project Management

13

Overview of ISO 9001

- The first version of ISO 9001 was published in 1987
- Versions of ISO standards are defined by the year of publications (e.g. ISO 9001:1994)
- Since software production is largely a question of design, ISO 9001 is the standard to use
- Its title is "Quality systems – Model for quality assurance in design, development, production, installation, and servicing"

Software Project Management

14

Overview of ISO 9001 (cont'd)

- ISO 9001 focuses on management instead of products
- Two basic requirements of ISO 9001
 - All operations influencing quality shall be under control
 - This control shall be visible (i.e. requires that plans, procedures, and organization be documented, and important activities be recorded)

Software Project Management

15

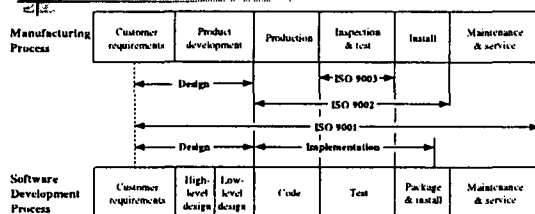
Overview of ISO 9001 (cont'd)

- ISO 9001 expects a fairly strict organization, where managers have the responsibility and authority to control the work of their subordinates (hence, self-organizing groups are difficult to fit into ISO 9001)
- Because ISO 9001 is written for the manufacturing industry, some interpretation is required to apply it to software development

Software Project Management

16

Software Development vs Manufacturing



Application of ISO 9001 Standard to the Manufacturing and Development Processes

Software Project Management

17

Three Levels of Quality Assurance

- ISO 9001 Quality systems – Model for quality assurance in design/development, production, installation, and servicing
 - If the software development organization designs the product it develops, then ISO 9001 will apply
- ISO 9002 Quality systems – Model for quality assurance in production and installation
 - If the software development organization implements products from a design that is provided to it, then ISO 9002 will apply

Software Project Management

18

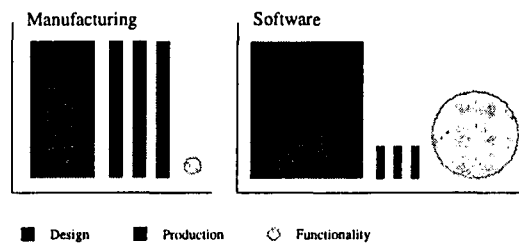
Three Levels of Quality Assurance (cont'd)

- ISO 9003 Quality systems – Model for quality assurance in final inspection and test
 - If the organization is a test organization, then ISO 9003 will apply
- Because ISO 9001 covers more aspects of development, more elements of the standard apply to ISO 9001 than to ISO 9002 and ISO 9003

Software Project Management

19

Manufacturing Industry vs Software Industry



Software Project Management

20

Manufacturing Industry Vs Software Industry (cont'd)

- Manufacturing
 - Design is a relatively minor activity (e.g. ball pens)
 - Production cost for each manufactured item is notable
- Software development
 - Nearly 100% design
 - Production cost for each copy of the software is insignificant
 - The functionality of software is of orders of magnitude as compared to other ordinary appliances

Software Project Management

21

Twenty Quality Elements in ISO 9000

1. Management responsibility
 - You must clearly define the general responsibilities of a company's management, in terms of: (i) quality policy, (ii) organization, and (iii) management review
2. Quality system
 - You must establish, document, implement, and maintain a quality system that conforms with ISO 9000

Software Project Management

22

Twenty Quality Elements in ISO 9000 (cont'd)

3. Contract review
 - You must have procedures for ensuring that what is expected from you is adequately defined and documented and that you have the capability to satisfy the requirements
4. Design control
 - You must have procedures for controlling and verifying the design output to ensure that specified requirements will be met

Software Project Management

23

Twenty Quality Elements in ISO 9000 (cont'd)

5. Document control
 - You must have defined procedures to control all documents, including review, approval, and change, and to ensure that the right level of information is available to the right people at the right time
 - You must also maintain a master list of current documents

Software Project Management

24

Twenty Quality Elements in ISO 9000 (cont'd)

- 6. Purchasing
 - You must ensure that parts, obtained from elsewhere, used in the product or in the production of the product, meet their specified requirements
- 7. Customer-supplied products
 - You must have procedures for verification, safe storage, and maintenance of products, or parts, provided by the customer to be included in the product

Software Project Management

25

Twenty Quality Elements in ISO 9000 (cont'd)

- 8. Product identification and traceability
 - Where appropriate, you must have procedures for identifying and tracing the product during all stages of production, delivery, and installation
- 9. Process control
 - You must carry out production under controlled conditions, including monitoring progress, approval of processes and equipment, etc.

Software Project Management

26

Twenty Quality Elements in ISO 9000 (cont'd)

- 10. Inspection and testing
 - You must have procedures for all levels of inspection and testing that you have identified as being required
 - You are also required to maintain records of test activity
- 11. Inspection, measuring, and test equipment
 - You must control, calibrate, and maintain inspection, measuring, and test equipment

Software Project Management

27

Twenty Quality Elements in ISO 9000 (cont'd)

- 12. Inspection and test status
 - You must be able to identify the test status of the product throughout the process
- 13. Control of nonconforming products
 - You must have procedures for controlling a product that does not conform to its specified requirements

Software Project Management

28

Twenty Quality Elements in ISO 9000 (cont'd)

- 14. Corrective action
 - You must have procedures for investigating the causes for nonconforming products and ensuring corrective actions to prevent recurrences
- 15. Handling, storage, packaging, and delivery
 - You must have a good system for storing and controlling the various parts that will compose your product during product development and through product delivery

Software Project Management

29

Twenty Quality Elements in ISO 9000 (cont'd)

- 16. Quality records
 - You must identify and keep records to demonstrate achievement of product quality and effective operation of your quality system
- 17. Internal quality audits
 - You must plan and carry out internal quality audits, by qualified individuals, to verify you are doing what you say you are doing and to determine the effectiveness of your quality system

Software Project Management

30

Twenty Quality Elements in ISO 9000 (cont'd)

- 18. Training
 - You must identify the training needs of your people, provide the required training, and keep records of the training
- 19. Servicing
 - You must have procedures for servicing your product when this requirement is specified in the contract
- 20. Statistical techniques
 - You must show that any statistical techniques that you use are correct

Characteristics of an ISO 9000 Quality System

- Quality objectives
 - The company should have a quality policy that states its quality goals and objectives and the strategy it will use to achieve them
- Commitment, involvement, and attitude
 - All employees and managers must be committed to the quality objectives and involved in achieving the objectives

Characteristics of an ISO 9000 Quality System (cont'd)

- Controlled
 - Every aspect of what is done during the development process must be controlled
- Effective
 - It is the means by which you measure whether your quality system is really working for you
- Auditable
 - ISO 9000 requires that systematic internal audits of your quality system be conducted

Characteristics of an ISO 9000 Quality System (cont'd)

- Documented quality system
 - Your quality system, including your processes and procedures, should be documented to the extent that, if you had to replace all of your employees, you could do it and still continue your business
- Continual improvement
 - ISO 9000 requires that your quality system be continually monitored and reviewed for weaknesses and that improvements be identified and implemented

Satisfying ISO 9000

- Quality policy
 - You must have a quality policy in written form
- Quality manager
 - You must assign a management representative, reporting at a high level, to be responsible for your quality system and for assuring ISO 9000 conformance
- Quality manual
 - ISO 9000 requires that your quality system be documented

Satisfying ISO 9000 (cont'd)

- Documented processes and procedures
 - You should document all procedures that would be needed to continue your operation if all of your people were replaced
- Project plan
 - For software development, this means planning the steps and activities that will be performed in transforming the product requirements into a final product

Satisfying ISO 9000 (cont'd)

- Build plan
 - It should specify what parts have to come together to create the total product, in what order, when, and it should specify their interdependencies
- Test plan
 - Every project should have a test plan that is established at the beginning of the project and updated as the project progresses

Satisfying ISO 9000 (cont'd)

- Service plan
 - Every product should have a service plan stating the planned maintenance activities that will be performed after the product is delivered and who will perform the activities
- Quality records
 - Quality records are kept so that you can show that you have done what you said you were going to do

Satisfying ISO 9000 (cont'd)

- Training records
 - ISO 9000 requires that you are able to show that you assign qualified people to various tasks and that you identify and provide required training to your employees
- Internal quality system audits
 - Periodic planned internal audits of your quality system should be conducted by qualified personnel for the purpose of determining the effectiveness of your quality system and ensuring that planned activities and procedures are being followed

Satisfying ISO 9000 (cont'd)

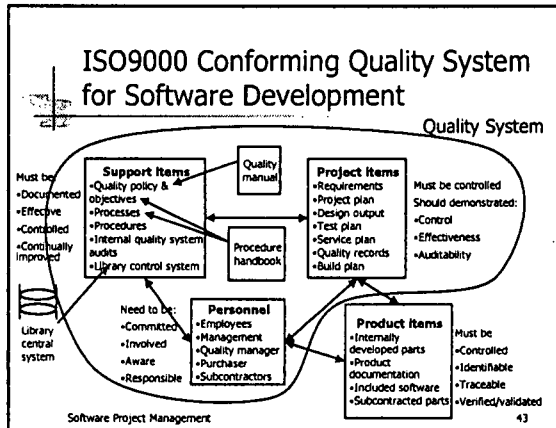
- Library control system
 - ISO 9000 requires proper and safe storage of the parts being developed
 - The library control system should also be used to store and control project and quality system documentation including, documented processes and procedures

Essentials Vs Standards Elements

| Essentials to conformance | ISO 9000 Standards Elements | | | | | | | | | | | | | | | | | | | |
|---------------------------------------|-----------------------------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| Quality objectives | X | X | | | | | | | | | | | | | | | | | | |
| Commitment, involvement, and attitude | X | | | | | | | | | | | X | | | | | | | | |
| Controlled | X | X | X | | | | X | X | X | X | | X | | | | X | X | | | |
| Effective | X | X | | | | | | | | | | | | | | | | X | | |
| Auditable | X | X | X | X | | | | | X | X | X | X | | X | X | X | | | | |
| Documented quality system | X | X | | | | | | | | | | | | | | | | | | |
| Continual improvement | X | | | | | | | | | | | | X | | | | | | | |
| Quality policy | X | | | | | | | | | | | | | | | | | | | |
| Quality manager | X | | | | | | | | | | | | | | | | | | X | |
| Quality manual | X | X | | | | | | | | | | | | | | | | | | |

Essentials Vs Standards Elements (cont'd)

| Essentials to conformance | ISO 9000 Standards Elements | | | | | | | | | | | | | | | | | | | |
|-----------------------------------|-----------------------------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| Documented procedures & processes | X | X | X | X | | | X | X | X | X | | | | X | X | X | X | X | X | X |
| Project plan | | | X | X | | | | | | | | | | | | | | | | |
| Build plan | | | | | | | | | | | X | X | X | X | | X | | | | |
| Test plan | | | X | | X | X | X | | | X | X | | | | | | | | | |
| Service plan | | | | | | | | | | | | | | | | | | X | X | X |
| Quality records | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| Training records | X | X | X | | | | X | | | | | | | | | | | | | X |
| Internal quality system audits | X | | | | | | | | | | | | | | | X | | X | | |
| Library control system | | | | | | | X | | | X | X | X | | | X | X | | | | |



Introduction of ISO 9000-3

- ISO 9001 is generic and many IT people feel it difficult to interpret and apply
- ISO 9000-3 is a set of guidelines that helps interpret and apply ISO 9001 for software development
- Since it is NOT a standard, companies are still assessed against ISO 9001

Software Project Management 44

Assumptions of ISO 9000-3

- Each development project is associated with a life cycle with phases
- The software product produced is the result of a contractual agreement between a purchaser and a supplier

Software Project Management 45

Overview of ISO 9000-3

- It consists of 22 clauses that do not correspond directly with the 20 clauses of ISO 9001
- These 22 clauses are grouped into three major sections:
 - Section 4: Quality system – Framework
 - Section 5: Quality system – Life cycle activities
 - Section 6: Quality system – Supporting activities

Software Project Management 46

Cross-reference ISO9000-3 to ISO9001

| Clause in ISO 9000-3 | Clause in ISO 9001 |
|------------------------------------|--------------------|
| 4.1 Management responsibility | 4.1 |
| 4.2 Quality system | 4.2 |
| 4.3 Internal quality system audits | 4.17 |
| 4.4 Corrective action | 4.14 |

Software Project Management 47

Cross-reference ISO9000-3 to ISO9001 (cont'd)

| Clause in ISO 9000-3 | Clause in ISO 9001 |
|---|-----------------------|
| 5.2 Contract review | 4.3 |
| 5.3 Purchaser's requirements specification | 4.3, 4.4 |
| 5.4 Development planning | 4.4 |
| 5.5 Quality planning | 4.2, 4.4 |
| 5.6 Design and implementation | 4.4, 4.9, 4.13 |
| 5.7 Testing and validation | 4.4, 4.10, 4.11, 4.13 |
| 5.8 Acceptance | 4.10, 4.15 |
| 5.9 Replication, delivery, and installation | 4.10, 4.13, 4.15 |
| 5.10 Maintenance | 4.13, 4.19 |

Software Project Management 48

Cross-reference ISO9000-3 to ISO9001 (cont'd)

| Clause in ISO 9000-3 | Clause in ISO 9001 |
|---------------------------------------|---------------------------|
| 6.1 Configuration management | 4.4, 4.5, 4.8, 4.12, 4.13 |
| 6.2 Document control | 4.5 |
| 6.3 Quality records | 4.16 |
| 6.4 Measurement | 4.20 |
| 6.5 Rules, practices, and conventions | 4.9, 4.11 |
| 6.6 Tools and techniques | 4.9, 4.11 |
| 6.7 Purchasing | 4.6 |
| 6.8 Included software product | 4.7 |
| 6.9 Training | 4.18 |

TickIT Initiative

- A system for certifying software development organizations to ISO 9001
- Led by the TickIT project office of the UK Department of Trade and Industry, and supported by the British Computer Society

TickIT Initiative (cont'd)

- Objectives of TickIT:
 - To ensure that the ISO 9000 series of standards is applied appropriately to software
 - To ensure consistency of certification within the IT industry
 - To enable mutual recognition of registration across the IT industry

TickIT Initiative (cont'd)

- TickIT scheme requires auditors to use the TickIT Guide (which is based on ISO 9000-3)
- The TickIT Guide tends to suggest more of how to implement an ISO 9000 conforming quality system than do the standards
- Under the TickIT scheme, auditors are required to pass a rigid set of criteria to become TickIT accredited

TickIT Initiative (cont'd)

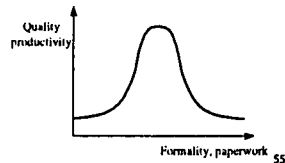
- TickIT auditors use ISO 9000-3 as a guide to check the quality system implemented in an organization
- If any discrepancy between the quality system and ISO 9000-3 is found, then these auditors will require explanations as to how the standards are being satisfied

Why Comply with ISO 9001?

- Provide a foundation for a quality system which is needed for quality software
- Increase productivity and reduce costs because development is done right the first time under control
- Ensure consistency of software quality
- Stay competitive by keeping up with market standards
- Fulfil software contractual requirements
- Improve corporate image

Potential Problems of ISO 9001

- Creating rules and formality to fulfill ISO 9001:
 - Too many rules result in bureaucracy
 - Too few rules result in insufficient control over quality



Software Project Management

55

Summary

- Quality is an elusive topic; we have problems:
 - defining it
 - achieving it
 - measuring it
- ISO 9000 provides an internationally mandated attempt to define and provide for (software) product quality in the customer-supplier relationship

Software Project Management

56

Summary (cont'd)

- Three important things about ISO 9000:
 - It is a tool for buyers, not builders
 - It is about what, not how
 - It provides necessary, but not sufficient, direction

Software Project Management

57

References

- Oskarsson, Ö., and Glass, R. L. (1996) *An ISO 9000 Approach to Building Quality Software*, Prentice Hall.
- Schmauch, C. H. (1994) *ISO 9000 for Software Developers*, ASQC Quality Press, Wisconsin.
- Dalfonso, M. A. (1996) *ISO 9000: Achieving Compliance and Certification 1996 Supplement*, Wiley.

Software Project Management

58

Software Project Management

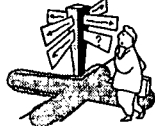
Lecture 7A
SEI - Capability Maturity Model

Overview

- A short history
- Software Process
- What is CMM - a detailed introduction
- Differences from ISO 9001

History

- In the 1980s, realization about the inability to manage the software process
 - Projects late, over budget, or plain failures
- 1986-1987: Software Engineering Institute (SEI)
 - Began developing a process maturity framework
- 1991: CMM-SW 1.0
- 1993: CMM-SW 1.1



What is a process? ...

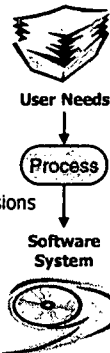
- It can be seen as a method that can be used to focus the efforts of a development team towards a desired result
- A process integrates:
 - People (with Knowledge, Skills, Training, Motivation)
 - Tools and Equipment
 - Procedures and Methods defining the relationship of tasks
- A good process will provide clear guidance, is disciplined and constantly refined based on experience

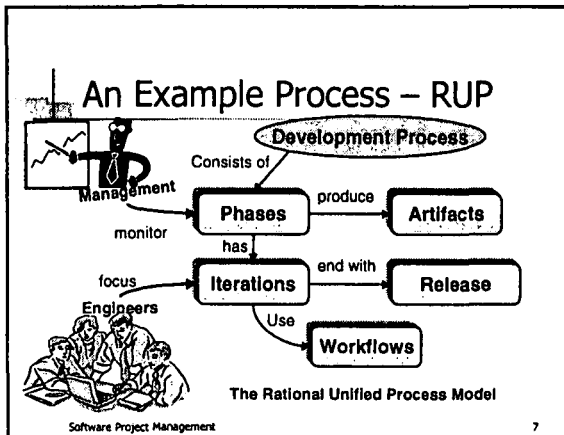
What is a process?

- A process provides a framework to work in:
 - Most developers take pride in their work and want to deliver quality output
 - Wrong tools that do not fit into the process can end up as shelfware
 - Procedures and Methods that do not support people will cause resistance – not ideal
- A good process resolves these issues and is flexible
- Processes are not there just to be followed, they are supposed to help deliver a better product (at lower cost)

How is a process used?

- Help ensure production of high-quality software matching the needs of the end-users
- Enhance team productivity
- Make purchase/hiring/management decisions
- Control schedule and budget
- Put software development best practices in action
- A well understood "Process Model" is used to communicate the details visually.





- ### Immature Organizations
- Immature Organization:
 - A defined/documentated process may not exist
 - If Processes exist they are improvised (as required), not rigorously followed
 - Managers react to crises only (fire fighting)
 - Ad-hoc project planning (poorly documented)
 - Schedules/budgets are rarely met (poor estimation)
 - Product quality is difficult to predict or judge
 - Difficult to maintain the products in the long term
 - Has a high turn-over of employees
- Software Project Management 8

- ### Mature Organizations
- Mature Organization:
 - Well-defined and well-followed processes that are updated when necessary (Process changes are formal)
 - Well-defined roles and responsibilities (Reduces confusion)
 - Product and process quality are monitored
 - Schedules are realistic (refined estimation process)
 - Participants understand value of the process (Staff are fully trained in the company process, expectations)
 - The deliverables from these organizations take longer, but the output is stable and predictable
 - Long term costs are low
- Software Project Management 9

- ### Process Management – A Premise
- "Improvements in process will improve quality"
 - This has been proven to work when the process is tuned to work with the people, tools, domain.
 - The process has to be defined within the context of available resources [people and money] as well as the deadline.
 - Total Quality Management principles have been shown to provide great benefits in manufacturing and service industries
 - Software is different, but some principles have been shown to work – CMM was built on top of these
- Software Project Management 10

- ### What is CMM?...
- Capability Maturity Model (CMM) is a framework that describes the key elements of an effective software process.
 - It describes an evolutionary improvement path from an ad hoc, immature process to a mature, disciplined process.
 - Covers practices for:
 - Planning
 - Engineering
 - Managing software development and maintenance.
 - When followed, these key practices improve the ability of organizations to meet goals for cost, schedule, functionality, and product quality.
-
- Software Project Management 11

- ### What is CMM?
- Establishes a yardstick against which it is possible to judge, in a repeatable way, the maturity of an organization's software process and compare it to the state of the practice of the industry [Kitson92]
-
- Software Project Management 12

Definitions from the CMM Specification

- We shall look at the definitions of:
 - Capability Maturity Model (CMM)
 - Software process
 - Software process capability
 - Software process performance
 - Software process maturity
- All definitions are quoted from the SEI CMM v1.1 Specifications.

Software Project Management

13

CMM - Definition

- A description of the stages through which software organizations evolve as they define, implement, measure, control, and improve their software processes
- A guide for selecting process improvement strategies by facilitating:
 - determination of current process capabilities
 - identification of the issues most critical to software quality and process improvement

Software Project Management

14

Software Process

- A *software process* can be defined as a set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products
 - E.g., project plans, design documents, code, test cases, and user manuals.
- As an organization matures, the software process becomes better defined and more consistently implemented throughout the organization.

Software Project Management

15

Software Process Capability

- *Software process capability* describes the range of expected results that can be achieved by following a software process.
- The software process capability of an organization provides one means of predicting the most likely outcomes to be expected from the next software project the organization undertakes.

Software Project Management

16

Software Process Performance

- *Software process performance* represents the actual results achieved by following a software process.
- Software process performance focuses on the results achieved, while software process capability focuses on results expected.

Software Project Management

17

Software Process Maturity

- *Software process maturity* is the extent to which a specific process is explicitly defined, managed, measured, controlled, and effective.
- Maturity implies a potential for growth in capability and indicates both the richness of an organization's software process and the consistency with which it is applied in projects throughout the organization.

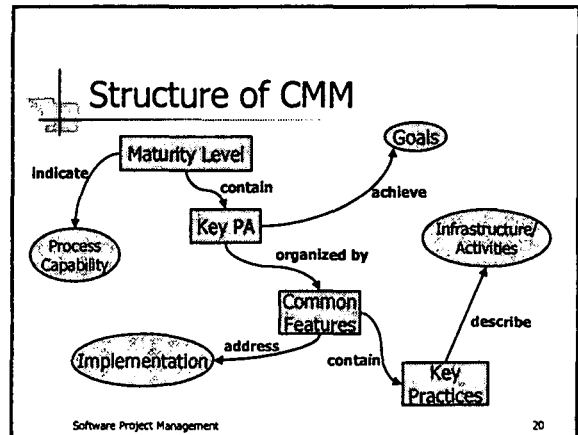
Software Project Management

18

Structure of CMM

- The CMM is composed of five maturity levels.
- Each maturity level is composed of several key process areas (except Level 1).
- Each key process area is organized into five sections called common features.
- The common features specify the key practices that, when collectively addressed, accomplish the goals of the key process area.

Software Project Management 19



Maturity Levels

- A maturity level is a well-defined evolutionary plateau toward achieving a mature software process.
- CMM provides for 5 top-levels:
 - Initial
 - Repeatable
 - Defined
 - Managed
 - Optimizing

Software Project Management 21

Maturity Levels

- Initial
 - No process, Ad-hoc response
- Repeatable
 - Disciplined Process
- Defined
 - Standard, Consistent Process
- Managed
 - Predictable Process
- Optimizing
 - Continuous Improvements

Software Project Management 22

Maturity Levels - Initial

- No stable environment for developing and maintaining software
- Difficulty making commitments that the staff can meet
- Crises are common
- Projects typically abandon planned procedures and revert to coding and testing
- Success depends entirely on exceptional managers and seasoned, effective software team
- Capability is a characteristic of the individuals, not the organization

Software Project Management 23

Maturity Levels - Repeatable

- Policies for managing a software project exist
- Procedures and Standards are defined
- Planning and managing new projects is based on experience with similar projects
- Basic software management controls exist
- Realistic project commitments based prior knowledge
- The software project managers track costs, schedules, and functionality; problems in meeting commitments are identified when they arise

Software Project Management 24

Maturity Levels - Defined

- Based on a common, organization-wide understanding of the activities, roles, and responsibilities in a defined software process
- The organization exploits best practices
- Special group responsible for software process
- Defined software process integrating engineering and management processes
- Management has good insight into technical progress on all projects
- Cost, schedule, and functionality are under control, and software quality is tracked

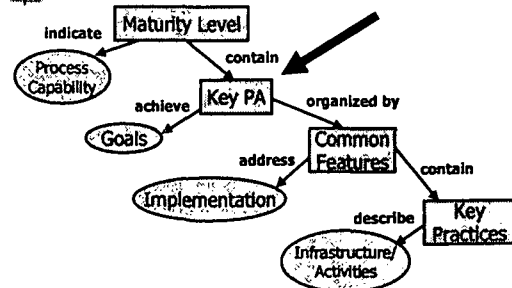
Maturity Levels - Managed

- Use of quantitative quality goals for both software products and processes
- Strong use of software/process metrics
- Organization-wide software process database is used
- Allows prediction of trends in process and product quality
- Process is both stable and measured
- Software products are of predictably high quality

Maturity Levels - Optimizing

- Organization level focus on continuous process improvement
- Innovations that exploit the best software engineering practices are identified and transferred throughout the organization
- Software processes are evaluated to prevent known types of defects from recurring
- Technology and process improvements are planned and managed as ordinary business activities

Structure of CMM



Key Process Area

- Each maturity level is composed of key process areas.
- Each key process area identifies a cluster of related activities that, when performed collectively, achieve a set of goals considered important for establishing process capability at that maturity level.
- The key process areas have been defined to reside at a single maturity level.
- For example, one of the key process areas for Level 2 is Software Project Planning.

Level 1 - Key Process Areas

- None that can be observed

Level 2 - Key Process Areas

- Software configuration management
- Software quality assurance
- Software subcontract management
- Software project tracking and oversight
- Software project planning
- Requirements management

Level 3 - Key Process Areas

- Peer reviews
- Inter-group coordination
- Software product engineering
- Integrated software management
- Training program
- Organization process definition
- Organization process focus

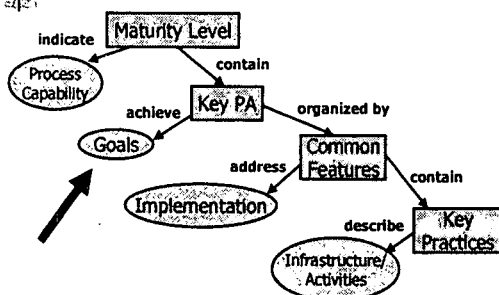
Level 4 - Key Process Areas

- Quality management
- Process measurement
- Software quality management
- Quantitative process management

Level 5 - Key Process Areas

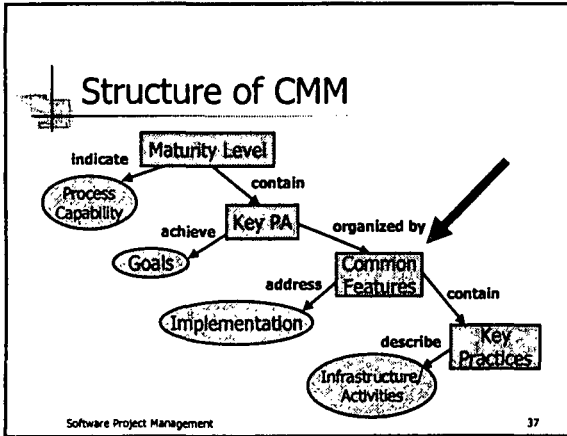
- Process change management
- Technology change management
- Defect prevention

Structure of CMM



Goals

- The goals:
 - Summarize the key practices of a key process area
 - Can be used to determine whether an organization or project has effectively implemented the key process area
 - Signify the scope, boundaries, and intent of each key process area
- E.g. : a goal from the Software Project Planning key process area :
 - "Software estimates are documented for use in planning and tracking the software project."



- ### Common Features
- The common features are attributes that indicate whether the implementation and institutionalization of a key process area is effective, repeatable, and lasting.
 - The key practices are divided among five Common Features sections:
 - Activities Performed (Describes Implementation Activities)
 - Commitment to Perform (Organizational culture)
 - Ability to Perform (Institutionalization factor)
 - Measurement and Analysis (Organization culture)
 - Verifying Implementation (Institutionalization factor).
- Software Project Management 38

- ### Common Feature - Commitment to Perform
- Describes the actions the organization must take to ensure that the process is established and will endure.
 - Typically involves establishing organizational policies and senior management sponsorship.
- Software Project Management 39

- ### Common Feature - Ability to Perform
- Describes the preconditions that must exist in the project or organization to implement the software process competently.
 - Involves resources, organizational structures, and training.
- Software Project Management 40

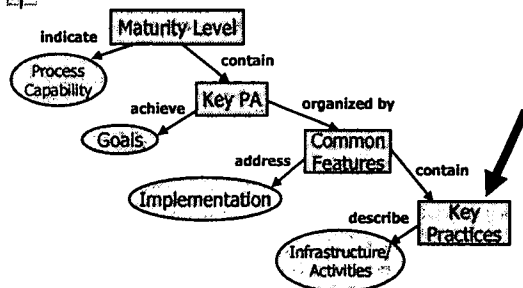
- ### Common Feature - Activities Performed
- Describe the roles and procedures necessary to implement a key process area
 - Cover what MUST be implemented to establish a process capability
 - Typically involve:
 - Establishing plans
 - Procedures
 - Performing the work
 - Tracking it
 - Taking corrective actions as necessary.
- Software Project Management 41

- ### Common Feature - Measurement & Analysis
- Describes the need to measure the process and analyze the measurements.
 - Typically includes examples of the measurements that could be taken to determine the status and effectiveness of the Activities Performed.
- Software Project Management 42

Common Feature - Verifying Implementation

- Describes the steps to ensure that the activities are performed in compliance with the process that has been established.
- Typically encompasses reviews and audits by management and software quality assurance.

Structure of CMM



Key Practices

- Each key process area is described in terms of key practices that, when implemented, help to satisfy the goals of that key process area.
- The key practices describe the infrastructure and activities that contribute most to the effective implementation and institutionalization of the key process area.
- For example, one of the practices from the Software Project Planning key process area is "The project's software development plan is developed according to a documented procedure."

Software Maturity – An Overview

- Just to sum up again:
 - Initial (No process, Ad-hoc response)
 - Repeatable (Disciplined Process)
 - Defined (Standard, Consistent Process)
 - Managed (Predictable Process)
 - Optimizing (Continuous Improvements)

ISO 9001 Vs CMM

- Almost all concerns raised by ISO 9001 are encompassed by CMM.
- ISO 9001 describes the minimum criteria for adequate quality management systems - rather than process improvement. CMM address process improvement as well as provides a clear path to achieving it.
- CMM provides more detailed guidance and greater breadth provided to software organizations.
- Building competitive advantage should be focused on improvement, not on achieving a score (which is the primary focus of ISO 9001).

ISO 9001 Certification Vs CMM Levels

- An ISO 9001-compliant organization would not necessarily satisfy all of the level 2 key process areas, it would satisfy most of the level 2 goals and many of the level 3 goals
- It is possible (in theory) for a level 1 organization to receive ISO 9001 registration
- A level 3 organization would have little difficulty in obtaining ISO 9001 certification
- A level 2 organization would have significant advantages in obtaining certification.



Quality Improvement Notes

- Enabling quality improvement is a management responsibility
- Quality improvement focuses on fixing the process, not the people:
 - However, the best chefs use the best ingredients
- Quality improvement must be measured:
 - All measurements must be "Goal" driven
- Rewards and incentives are necessary to establish and maintain an improvement effort:
 - Should fit into the organizations culture
- Quality improvement is a continuous process
- Not all problems are technical

Software Project Management

49



References

- Software Engineering Institute Website - <http://www.sei.cmu.edu>
- Paulk, M.C., Curtis, B., Chrissis, M.B., and Weber, C.V. (1993a) "The Capability Maturity Model for Software, Version 1.1", SEI Technical report CMU/SEI-93-TR-024.
- Paulk, M.C., Weber, C.V., Garcia, S.M., Chrissis, M.B., and Bush, M. (1993b) "Key Practices of the Capability Maturity Model SM, Version 1.1", SEI Technical report CMU/SEI-93-TR-025.

Software Project Management

50

Software Project Management

Lecture 8 Software Project Performance Tracking and Monitoring

Overview

- Importance of tracking and monitoring
- Creating a management framework
- Tracking the performance
- Monitoring the progress and resource
- Getting project on the right track

Software Project Management

2

Importance of tracking and monitoring

- Make sure the project
 - Can be delivered on time and within budget
 - Is of good quality
 - Meets client's needs

Software Project Management

3

What can go wrong in product?

- Inadequate functionality of a product
 - Related to SRS
- Poor quality of a product
 - Related to quality management
- Late delivery of the product
- Overly exceeding the budget

Software Project Management

4

Planning, Tracking and Monitoring

- Planning
 - Know where we want to go
- Tracking
 - Know where we are
- Monitoring
 - How to go from where we are to where we want to go

Software Project Management

5

Tracking

- Finding out what is happening
- Need a plan and schedule
- To collect data

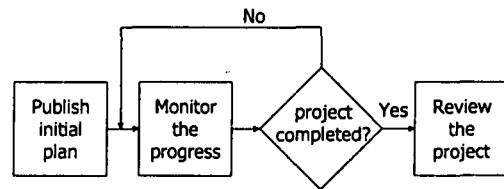
Software Project Management

6

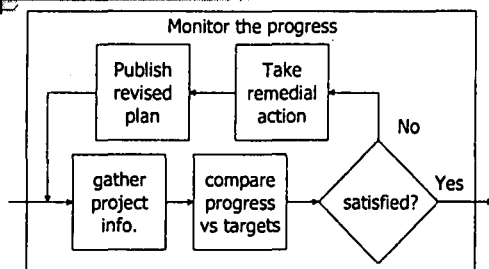
Monitoring

- Comparing the current status with the targets
- Need a plan, a schedule, collected data
- To exercise control over the project
- To ensure the targets are met
- To devise contingency plans

A suggested framework



A suggestion framework (cont'd)



Tracking the performance

- Setting check points
- Collecting data

Check Point

- Based on regular time intervals
 - Can be weekly or monthly or quarterly
 - Depend on what to check and how to
- Based on a particular event
 - At the end of each activity
 - In the middle of a critical activity

Check Point (cont'd)

- Should be set before the plan was published
 - Make sure everyone knows when and what the check points are

Collecting data

- Partial completion report
- Risk report

Partial completion report

- Indicate the work done by the personnel and the time spent on the work
- Optional items
 - likelihood of failing to complete the task by the scheduled date
 - Estimated time of completion

Partial completion report – Example

| Time Sheet | | | | | | |
|------------------------|---------------|----------------|-----------------------|----------------------------|-----------|-----------|
| Staff: Paul | | | Week ending: 14/05/99 | | | |
| Rechargeable hours | | | | | | |
| Project | Act. code | Desc. | Hours | % done | Sch. date | Est. date |
| P20 | A267 | Code mod A7 | 24 | 90 | 01/06/99 | 20/05/99 |
| P35 | B397 | Testing mod B8 | 12 | 30 | 24/06/99 | 24/06/99 |
| | | | Total | 36 | | |
| Non-rechargeable hours | | | | | | |
| Code | Desc. | | Hours | Comments and Authorization | | |
| L90 | hours in Lieu | | 4 | Authorized by Peter | | |
| | | | Total | 4 | | |

Risk reporting

- Indicate the likelihood of meeting the scheduled target date
 - Instead of asking the estimated completion date
- Use the traffic-light method

The traffic-light method

For assessing a product

- Identify the key (first-level) elements
- Break them into smaller components
- Assess each component by
 - Green as 'on target'
 - Amber as 'not on target but recoverable'
 - Red as 'not on target and recoverable only with difficulty'

The traffic-light method (Cont'd)

- Assess the key-level element based on the assessments of their components
- Assess the overall product based on all the assessments (key elements and their components)

The traffic-light method -- Example

| Activity Assessment Sheet | | | | | |
|---|----|----|----|----|----------|
| Staff : Zobel | | | | | |
| Ref:IoE/P/100 Activity: Code and test module A | | | | | |
| Week number | 13 | 14 | 15 | 16 | |
| Activity summary | G | A | R | | |
| Component | | | | | Comments |
| Screen handling procedures | G | G | A | | |
| File updating | G | A | R | | |
| Compilation | G | G | A | | |
| Run test data | G | A | A | | |

Software Project Management

19

Monitoring the progress

- Need to monitor time
- Need to monitor cost

Software Project Management

20

Monitoring the time

- Tools for visualizing the progress
 - Presenting the collected data in a way that is easy to understand
 - Help to easily identify the problem activities or areas that need to be taken care of

Software Project Management

21

Visualizing Techniques

- The Gantt chart
 - A static picture showing the current progress of the project
- The Timeline
 - A dynamic picture showing the progress of the project and how the project has changed through time

Software Project Management

22

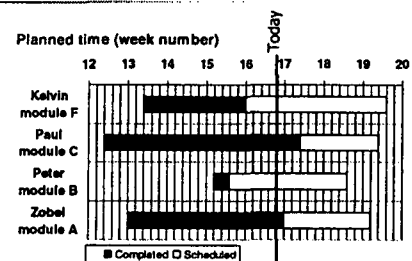
The Gantt Chart

- An activity bar chart showing
 - the activities, their scheduled dates and duration
 - the reported progress of the activities;
 - 'today cursor'

Software Project Management

23

The Gantt Chart (cont'd)



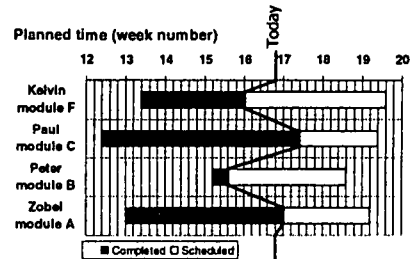
Software Project Management

24

The Slip Chart

- Add a slip line on the Gantt chart
- The slip line indicates those activities that are either ahead or behind the schedule
- Too much bending indicates a need for rescheduling of the overall plan

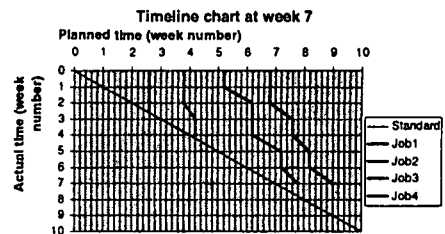
The Slip Chart (cont'd)



The Timeline

- A plot of the elapsed time against the planned time of the activities indicating
 - the actual progress of the activities; and
 - the rescheduled activities by the end of each week
- show where and when the targets have changed through the life of a project

The Timeline (cont'd)



The Timeline (cont'd)

- Can show the slippage of the activities through the life of the project
 - The Gantt chart cannot
- Help to analyze and understand the trends and reason for changes
 - to avoid slippage in future projects

Monitoring the Cost

- Earned Value Analysis
 - A cost monitoring technique recommended by DoD of US and Australia

Earned Value Analysis

- Produce a baseline budget from the project plan
 - Calculate the earned value of each activity
 - Earned value* = time for an activity / total time for the project
 - E.g. earned value = number of days for an activity / number of days for the project

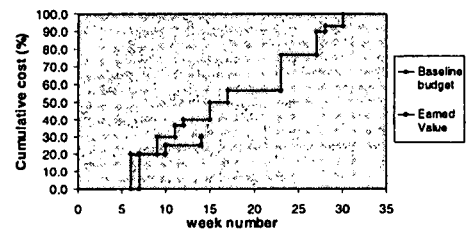
Earned Value Analysis (cont'd)

- Monitor the earned value
 - Once an activity is completed, its elapsed time is recorded and its earned value (EV) is accumulated to the cumulative EV

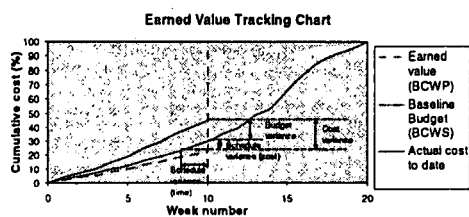
Earned Value Analysis (cont'd)

| Task | Budgeted week | Scheduled completion | Cumulative weeks | % cumulative earned value |
|------------------------|---------------|----------------------|------------------|---------------------------|
| Specify overall system | 6 | 6 | 6 | 20.0% |
| Specify module A | 3 | 9 | 9 | 30.0% |
| Specify module B | 2 | 11 | 11 | 36.7% |
| Check specification | 1 | 12 | 12 | 40.0% |
| Design module A | 3 | 15 | 15 | 50.0% |
| Design module B | 2 | 17 | 17 | 56.7% |
| Code and test module A | 6 | 23 | 23 | 76.7% |
| Code and test module B | 4 | 27 | 27 | 90.0% |
| System Integration | 1 | 28 | 28 | 93.3% |
| System Testing | 2 | 30 | 30 | 100.0% |

Earned Value Analysis (cont'd)



Earned Value Analysis (cont'd)



Earned Value Analysis (cont'd)

- Budget variance
 - = Actual cost to date - Baseline budget
 - Indicates how actual cost differs from the planned cost

Earned Value Analysis (cont'd)

- Schedule variance
 - = Earned Value – Baseline budget
 - Indicates how the actual schedule differs from the planned schedule
- Schedule performance index
 - = Earned Value / Baseline budget
 - SPI > 1 means "better than planned"
 - SPI < 1 means "slower than planned"

Software Project Management

37

Earned Value Analysis (cont'd)

- Cost variance
 - = Earned Value – Actual cost to date
 - Indicates how the planned cost differs from actual cost
- Cost Performance index, CPI
 - = Earned Value / Actual cost to date
 - CPI > 1 means "better than planned"
 - CPI < 1 means "slower than planned"

Software Project Management

38

Prioritizing Monitoring

- Priority list of activity to monitor
 - Critical activities
 - Non-critical activities with no free float
 - Non-critical activities with less than a specified float
 - High risk activities
 - Activities with critical resources

Software Project Management

39

Bringing the Project Back to Target

- You are now behind the schedule
- Possible actions:
 - Reschedule the target date
 - Reschedule other activities with shorter duration
 - Reorder the activities

Software Project Management

40

Shorten the Critical Activities

- Putting pressure on the personnel
- Increasing the resources
 - Personnel work longer hours
 - Additional analysts to interview users
 - Competent programmer to code modules in the critical activity

Software Project Management

41

Reorder the activities

- Relax the constraints on the start of an activity before the completion of the previous one
- Subdivide the components of an activity so that they can be done in parallel

Software Project Management

42



References

- Hughes, B., and Cotterell, M. (1999) *Software Project Management*, 2nd ed., McGraw Hill.
- Down, A., Coleman, M., and Absolon, P. (1994) *Risk Management for Software projects*, McGraw Hill.

Software Project Management

Lecture 9 Software Configuration Management

Overview

- In this lecture we shall cover,
 - Software evolution (types of changes)
 - Configuration management (need for it)
 - Facets of SCM
 - Change control board
 - Change management
 - Auditing and status accounting

Software Project Management

2

Introduction

- 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"

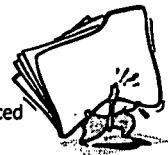


Software Project Management

3

Software Evolution

- Software evolves over a period of time
 - Many different items are produced over the duration of the project
 - Different versions are produced
 - Teams work in parallel to deliver the final product
- Software evolution implies a constantly changing system

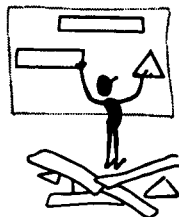


Software Project Management

4

How software changes...

- The four aspects of software evolution are:
 - Corrective changes
 - Adaptive changes
 - Perfective changes
 - Preventive changes



Software Project Management

5

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)

Software Project Management

6

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

Software Project Management

7

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

Software Project Management

8

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

Software Project Management

9

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

Software Project Management

10

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

Software Project Management

11

So what is the answer??

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



Software Project Management

12

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

Configuration Management

- 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

Facets of SCM

- The four core aspects of SCM are:
 - Configuration identification
 - Configuration control and change management
 - Configuration auditing
 - Status accounting

Configuration Identification...

- A large number of items are part of the software development process:
 - Source and binary modules
 - Hardware and operating systems
 - Documentation
 - Requirements
 - Design
 - Test cases
 - Etc..
- Key is to identify the items that need to be under SCM

Configuration Identification...

- Configuration identification is the process of establishing a baseline from which system changes are made – allows for control.
- So what needs to be under SCM?
 - Items where changes need to be tracked and controlled
 - If in doubt, add it into the inventory of items under SCM
- Common items under SCM are:
 - Source code, documentation, hardware/OS configuration

Software Project Management

19

Terminology Review – 1

- **Configuration items** - any single atomic item for which changes need to be tracked
 - Source code file
 - The project plan
 - The documentation standard
 - ...
- **Baseline** - A product that has been formally approved, and consists of a well-defined set of consistent configuration items

Software Project Management

20

Terminology Review - 2

Baseline

"A specification or product that has been formally reviewed and agreed to by responsible management, that thereafter serves as the basis for further development, and can be changed only through formal change control procedures"

[Bruegge]

Software Project Management

21

Baseline Types

- As the system is developed a number of baselines are created:
 - *Developmental baseline* (RAD, SDD, integration test, ...).
 - Goal: coordinate engineering activities.
 - *Functional baseline* (prototype, technology preview, alpha, beta release).
 - Goal: obtain customer experiences with functional system.
 - *Product baseline* (GA with a version - win95, word 2000).
 - Goal: coordinate sales and customer support.

Software Project Management

22

Managed Directories

- Programmer's directory (IEEE: dynamic library).
 - Library for holding newly created or modified software entities. The programmer's workspace is controlled by the programmer only.
- Master directory (IEEE: controlled library).
 - Manages the current baseline(s) and for controlling changes made to them. Entry is controlled, usually after verification. Changes must be authorized.
- Repository (IEEE: static library).
 - Archive for the various baselines released for general use. Copies of these baselines may be made available to requesting organizations.

Software Project Management

23

Configuration Identification

- A few notes...
 - Starting too early can add too many items that may really not require full configuration management.
 - Starting too late will result in a disaster
 - It is common to have 1000+ items under SCM
- A good start
 - Place all documents under SCM
 - Add code as it starts to be available
 - Remove older items and archive them
 - Remove items where the changes are minor, rare and need not be under the purview of a complete SCM

Software Project Management

24

Version Allocation...

- Once a configuration item (CI) has been identified – a proper version number must be allocated
- The best option is to start with a major-minor versioning scheme
 - Major version numbers are between 0 – n
 - Minor version numbers should be between 0 – 100

Version Allocation...

- Examples:
 - Report.Java (version 1.23)
 - Major version: 1.0
 - Minor version: 23 (indicative of number of revisions to this file)
 - Project plan (version 6.34d)
 - Major version: 6
 - Minor version: 34
 - The "d" is indicative of "draft"
- Versioning scheme is developed by the company to suite their needs

Version Allocation...

- Often many companies prefix the configuration item based on its type.
 - Documentation may be prefixed "doc"
 - Source code can be "src"
 - Example: doc-pmp-2.34
 - Project management plan document (version 2.34)

Version Allocation

- New versions of software can be:
 - Maintenance releases
 - Minor upgrades
 - Technology refresh or major upgrades
 - Technology insertion

Baseline Levels

- The software system can be tagged at various stages of its evolution with a baseline number
 - **Development baseline** "n" (where the "n" can be indicative of the 10% of functionality implemented)
 - **Testing baseline** (where a specific build is created for the specific purpose of testing)
 - **Release baseline** (where the software is built for GA)
- There is no rule on when to baseline – but a good guideline is to have one a week

Terminology Review – 3

- **Version** – an *initial* release or re-release of a configuration item (*ideally different versions should have different functionality*)
- **Revision** – minor *changes* to a version that correct errors in the design/code (*typically revisions do not affect expected functionality*)
- **Release** – the *formal distribution* of a **baseline**

Configuration Control and Change Management

- Review of change activity can highlight what is changing and what is not.
 - Impact of change can be measured over time
- Issues to consider are:
 - Keeping track of changes (deltas or separate files)
 - Allows for parallel development on a single item (many developers updating the same file)

Software Project Management

31

Deltas Vs Separate Files

- After the initial baseline has been established – the item is said to be under SCM.
- Changes can be tracked as:
 - Deltas: *only the changed portion is stored*
 - Separate file: *changes are stored in a new file*
- Deltas work best for text files
- Separate files is a good idea for binary file formats.

Software Project Management

32

Parallel Development

- Many members can often work on the same item:
 - Two developers update the same code file (working on different functions)
 - A number of engineers may be working on a single word document containing the specifications
- Changes are tracked by each user and often merged regularly to create a synchronized version:
 - Merge conflicts are resolved via normal channels of communications
 - Effective management can reduce merge conflicts

Software Project Management

33

Change Management – 1

- For best results changes should be handled formally
 - A change control board (CCB) is necessary
- CCB consists of all key stakeholders
 - Customers
 - Developers
 - Designers and architects
 - Management
 - Business strategists and financiers

Software Project Management

34

Change Request

- Changes are required because:
 - A problem is discovered (bug?)
 - An enhancement is required
- Once a change is required – a “change request” is raised
- A change request (CR) will outline:
 - Current operation, nature of problem/enhancement, expected operation after system is changed

Software Project Management

35

Change Control Board – 1

- All change requests (CR) are reported to the CCB for review
- CCB discusses all open CRs at regular meetings (frequency is determined by nature of project)
- CCB determines if CR identifies a “problem” or an “enhancement”
 - This is done to identify who “pays” for the change

Software Project Management

36

Change Control Board – 2

- Once the change has been categorized, it is discussed in detail,
 - Probable source of problem
 - Impact of the change
 - Time and resource requirements (estimates)
- CCB will assign a priority and severity for all CRs (CRs may also be rejected)
- The CRs are assigned to development management for further action

Impact Analysis

- Before changes are made often a deep or shallow impact analysis is performed
- Impact analysis makes full use of software metrics
- Managers often track
 - Increases in complexity measures as system evolved over a period
- Trend analysis is performed based on modules and change requests to ensure flexibility

Change Management – 2

- Development managers will assign a CR to a developer (or a team)
- The requested change is made as per the plan and a full regression test suite is executed
- Configuration manager reviews the changed system
 - Ensures that all required documentation is changed
 - Ensures that the impact does not exceed estimates (too much)

Configuration Management

- To ensure proper tracking the following information needs to be collected:
 - When was the change made
 - Who made the change
 - What was changed (items modified)
 - Who authorized the change and who was notified
 - How can this request be cancelled
 - Who is responsible for the change
 - Priority and severity
 - How long did the change take (vs estimate)

Change Management – 3

- Changes will need to be modified or cancelled
 - This is required if the team assigned the change request need more resources and time than estimated
- These decisions are delegated to CCB with the extra information added along with the CR.
- A CR can be *assigned, deferred, rejected, closed*.
- All changes are reviewed and modified

Change Control Board – 3

- The complexity of the change management process varies with the project
- Small projects can perform change requests informally
- Complex projects require detailed change request forms and the official approval by one or more managers
- For safety critical projects – change management is very rigorous
 - Example: software changes to an airplane avionics system
- Change request management is supported by robust software packages

Change Management – 4

- To ensure effective change management:
 - Each working version is assigned an identification code
 - As a version is modified, a revision code or number is assigned to each resulting changed component
 - Each component's version and status as well as a history of all changes are tracked

Configuration Auditing – 1

- Key philosophy is "trust by verify"
- Configuration auditing is a process to:
 - Verify that the baseline is complete & accurate
 - Check that changes made and recorded
 - Documentation reflects updates
- Audits can be rigorous, or on a random set of configuration items
- A regular audit is required to ensure that SCM is working efficiently:
 - Can reveal weaknesses in processes, tools, plans as well as resources

Configuration Auditing – 2

- The two main types of audit are:
 - **Physical audit:** *are all identified items have a correct version and revision, this helps us remove old and unnecessary items.*
 - **Functional audit:** *verifies that the items under SCM satisfy defined specifications.*

Status Accounting

- Simple record that can identify all items under SCM
 - Location of the component, who placed it under SCM etc
 - The current version
 - Revision history (change log)
 - Pending CRs
 - Impact analysis trends
- Status accounting is vital to enable control and effective management

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

Roles and Responsibilities...

- 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

Standards

- Approved by ANSI:
 - IEEE 828: software configuration management plans
 - IEEE 1042: guide to software configuration management

Outline of the SCMP (IEEE 828-1990)

- 6 main sections:
 - Introduction
 - Management
 - Activities
 - Schedule
 - Resources
 - Maintenance

Conformance to the IEEE Standard 828-1990...

- A document titled "software configuration management plan" is required.
- Minimum required sections are:
 - Introduction
 - Management
 - Activities
 - Schedule
 - Resources
 - Plan maintenance

Conformance to the IEEE Standard 828-1990

- Quality guidelines:
 - All activities defined in the plan must be assigned
 - All configuration items should have a defined process for establishing the baseline and change control
- If these criteria are met, the SCMP can state:
"This SCMP conforms with the requirements of IEEE Std. 828-1990."

Tools

- A large number of tools are available
- Examples:
 - RCS – pessimistic locking control system, no support for parallel development
 - CVS – based on RCS, allows concurrent working using optimistic locking
 - Star team – multiple servers, process modelling, policy check mechanisms.

Repositories

- All items under configuration are placed into a repository
- This repository is controlled by a tool like RCS or CVS
- A CVS (RCS as well) repository can handle both binary and text files
 - Changes are stored as deltas for text files
 - Separate files are stored for binary files

Functions of a Repository

- Access to repository is controlled by a security policy (in CVS/RCS a username/password)
- After a user is logged into the repository they can:
 - Check-out a file for use
 - Check-in a changed file back into the repository
 - Tag the repository at a certain date/time
 - Place a new file into the repository

File Check-Out

- Check-out:
 - The repository marks the file as checked-out
 - In concurrent systems (like CVS), a list of all users that have checked out the file is maintained
 - In locking systems (like RCS), only one user can check-out a file at a time. In essence the file is locked for future use
- CVS can work in both locking and concurrent modes

File Check-In...

- Check-in:
 - The file is modified and the changed version is checked back into the repository
 - If the file was locked – then only the user that checked it out can unlock it by checking in the same file (verified on file name only)
 - In a concurrent system – the first user to check in will cause the system to enter a conflict resolution mode

File Check-In...

- In a concurrent versioning system:
 - Users can potentially work on the same file at the same time
 - If two users change the same sections of the file then when checking in, the system will flag a merge conflict
 - Merge conflicts have to be resolved before the file can be placed into the repository
 - This allows more flexibility, however it can be more dangerous as well when used without sufficient training

File Check-In

- Most companies have procedures that will outline the steps to take before a file is checked back into the repository
- All changes made to the file are documented in a "status log" during the check-in process
 - Some procedures require the change request number to be stated after a certain date

Ensuring Build Consistency

- When developers check-in source code with modifications – the changes may cause more bugs
- To ensure that new changes do not cause unexpected failures many techniques are used by developers:
 - Regression testing
 - Compile & link verification
 - Static audits
 - Metrics trends
- This aspect of development is one of the most important and requires careful monitoring

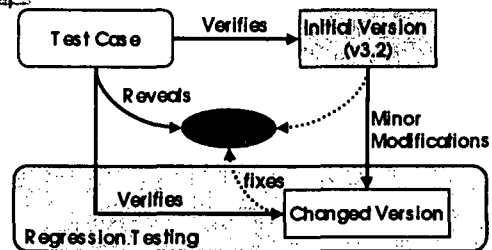
What is Regression Testing?

- Simply put, it is repetition of existing tests
 - Usually done after minor changes are made to code
 - It does not apply for enhancements
 - Before checking in the changes into a repository
- It can be seen as selective testing
- Intention is to show that modifications have not caused unintended effects
- Verifies that the system still complies with its specified requirements

Software Project Management

61

Regression Testing



Software Project Management

62

Regression Testing Limitations

- Regression test suite does not contain tests for new or changed capabilities
- When a primary test suite is promoted to become a regression test suite, it is no longer effective as a primary test suite
 - Once a version has passed all of its test cases, the test suite has revealed all the bugs that it can and must be changed to look for new changes

Software Project Management

63

Compile and Link Verification

- This process is applied mainly to source code before checking it into a repository
 - The changed copy is built locally on the developers workspace
- The aim is to ensure that there are no compiler errors, warnings or link failures when integrated with the existing set of code as mirrored on the repository
- This is the simplest form of check, and increases confidence – this does not check for bugs.

Software Project Management

64

Static Audits

- Applicable to source code
- Before check-in the changed code is passed through a static code verification tool
- Any violations or failure to meet company standards are picked up by this tool
- All issues are resolved before check-in
- This step can ensure that the overall quality of the code in the repository is higher
- Does not detect functional bugs or errors that can be caused at runtime.

Software Project Management

65

Metrics Trends

- This step is another step to ensure that poor quality code does not enter the repository
- Company can define a global standard on the size of methods, allowed complexity etc...
 - If changes have caused a deviation from the norm, then it required approval from the team leader before being checked into the repository
- The code is analyzed over a period to ensure that over time the code does not deteriorate.

Software Project Management

66

Tagging a Repository...

- As software evolves the files that make up the system are all at different version levels
- Example (from an O/S can be)
 - Text editor v3.0
 - Internet explorer v5.5
 - Kernel build 1885
 - Windowing system build 23
 - Etc...
- Before the above configuration is released – we need to note down these numbers. That process is known as tagging the repository

Software Project Management

67

Tagging a Repository

- Once a repository is tagged
 - We can revert back to this level even if a large number of changes are made
 - The tag number can be issued to the quality and testing teams for verification.
- Repository is tagged at regular intervals,
 - Weekly is typical early in the life cycle
 - Daily towards the end or if the process calls for high-frequency integration

Software Project Management

68

Keywords

- Configuration item
- Version
- Baseline
- Release
- Revision
- Change management
- Configuration management

Software Project Management

69

Summary

- SCM is an insurance policy.
- Effective SCM can ensure:
 - Rework cost is reduced
 - Effort put into development is not wasted
 - There is no loss of control as software evolves

Software Project Management

70

References

- Ghezzi, C., Jazayeri, M., and Mandrioli, D. (1991) *Fundamentals of Software Engineering*, Prentice Hall
- Horch, J. W. (1996) *Practical guide to software quality management*, Artech House.
- Bruegge, B., and Teubner, G., *Object-oriented software engineering: conquering complex and changing systems*
- IEEE standard 828-1990
- Lientz, B. P., and Swanson, E. B. (1981) "Problems in application software maintenance", *Communications of the ACM*, 24(11):763-769.

Software Project Management

71

Software Project Management

Lecture 10
Project Team
Management & Organization

Lecture Overview

- Motivation
- Organizational Behaviour Theory
- Organizational Planning
- Team Formation and Development
- Roles in Software Development

Projects – Temporary Nature

- The temporary nature of projects means that the personal and organizational relationships will generally be both temporary and new. The project management team must take care to select techniques that are appropriate for such transient relationships.

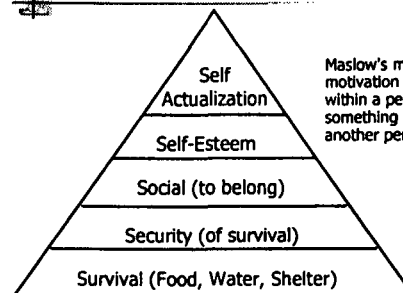
Motivation – Definition

motivate /-v 1. Supply a motive to; be the motive of 2. cause (a person) to act in a particular way 3. stimulate the interest (of a person in an activity) -
The Oxford Dictionary of Current English

Maslow's Hierarchy – 1

- Proposed a more "active" model of motivation.
- Redefined motivational theory.
- He argued that all people were driven or motivated by a hierarchical set of needs.

Maslow's Hierarchy - 2



Maslow's model places motivation as something within a person rather than something provided by another person

Herzberg's Theory - 1

- More focussed in organizations
- Two sets of motivational factors:
 - De-motivators or dissatisfiers
 - Higher Performance Motivators

Herzberg's Theory - 2

- Examples of De-Motivators:
 - Company Benefits or Culture
 - Job Security
 - Work Conditions
 - Salary [or perceived inequity of]
 - Interpersonal or Team relations
- Absence of de-motivating factors meant that people did "a fair days work".

Herzberg's Theory - 3

- Examples of Higher Performance Motivators:
 - Opportunity for growth
 - Nature of work tasks
 - Responsibility
 - Recognition
 - A sense of achievement
 - Autonomy
- These factors are some of the most crucial, however depending on experience and industry these have been shown to change.

Contemporary Theory

- Many variations and extensions to the work by Maslow and Herzberg have been made, but the fundamental validity remains unchallenged.
- Most contemporary motivation theory agrees that there are *intrinsic* and *extrinsic* motivation factors which align with Maslow's and Herzberg's initial work.

Motivation by Leaders

- A leader can motivate his or her people by addressing factors such as working conditions, physical environment, team interpersonal relations and basic rewards.
- The more powerful or high performance motivation needs are within each person and are controlled by each individual team member not the leader.
- To impact the high performance intrinsic motivation, a project manager/team leader must address the external factors such as autonomy, opportunity for growth and so on.

Hackman & Oldham Theory-1

- Hackman and Oldham analysed the intrinsic motivation impact of job and job tasks.
- They provided a structured model for improving intrinsic motivation.
- Proposed that each job contains a number of *core job dimensions*.

Job Diagnostic Model - 1

- States that positive outcomes occur when employees experience meaningfulness in their work.
- Meaningfulness occurs through:
 - Skill variety -- a person needs to use a number of different skills and talents.
 - Task variety -- their work involves completing a "whole" and identifiable piece of work.
 - Task significance -- the work has a substantial impact on the lives or work of other people.
 - Autonomy -- the person has freedom in accomplishing the tasks.
 - Feedback -- the job provides some built-in feedback or reward.

Software Project Management

13

Job Diagnostic Model - 2

- Their Job Diagnostic Survey consists of a series of questions that employees answer anonymously. Some questions are:
 - How much independence and freedom do you have in the way you carry out your work assignments?
 - How effective is your manager in providing feedback on how well you are performing your job?
 - To what extent does your job require you to use a number of complex or high-level skills?

Software Project Management

14

Hackman & Oldham Theory-2

- If the core job dimensions are improved then the person undertaking the job develops an internal belief in the meaningfulness of the job, responsibility for and understanding of the relationship between their effort and the results.
- These internal states lead to improved personal and work outcomes such as high intrinsic motivation, high quality work and so on.

Software Project Management

15

Motivational Approaches for IT Projects

- 3 levels of intrinsic motivation that apply for computer and other creative teams:
 - Technical Excellence
 - Client Partnership
 - Adding Value

Software Project Management

16

Motivational Approaches

- Simple approaches that work:
 - Share the client
 - Share the project vision
 - Share the skills and knowledge
 - Share the success
 - Deliver early and often
 - Raise the team profile

Software Project Management

17

Some rewards..

- Rewards that work in IT:
 - research and development "time-outs"
 - feedback from the business management
 - be there when it works
 - actively support extra-curricular activities
 - encourage fun and play
 - arrange for rotation
 - ask your team what rewards matter to them

Software Project Management

18

Workplace philosophies

- Management approaches:
 - Taylorism, 'Scientific Management'
 - Human Relations School
- Counter positions:
 - Braverman School
 - Human-Centred School

Software Project Management

19

Taylorism

- Assumes that people work only for anticipated extrinsic benefits...
Individual rewards:
 - Detailed breakdown of the labour process into minute detail
 - Separation of conception from execution
 - Progressive destruction of craft skills
 - No autonomy over pace, order, timing of tasks

Software Project Management

20

Human Relations School

- People have a need for moral satisfaction in their work... Group rewards:
 - Allow and encourage sociability among workers and work groups
 - Semi-autonomous work groups deciding:
 - how tasks are organised
 - who does what
 - when/how work is rotated
 - pace of work

Software Project Management

21

HR versus Taylorism

- HR claims that Taylorism is not effective, and that
 - HR is more efficient for work under any economic system
 - work is compatible with private ownership
 - ownership of means of production is not an issue
 - work autonomy is central

Software Project Management

22

Braverman School

- Taylorism is the 'managerial ideology of advanced capitalism', but
- Taylorism is the most effective form of class control over labour in the work place
- Equate economic triumph of capitalism with triumph of Tayloristic control
- Work can only be rewarding under socialist regime

Software Project Management

23

Human-Centred School

- Want to overturn Taylorist orthodoxy in design of work and technology
 - Retain centrality of human skills and intuitive knowledge
 - People should see the entire production process from end to end
 - Commitment comes from pride in skill and competence

Software Project Management

24

Summary of approaches - 1

- Taylorism advocates extreme division of labour, works under capitalism
- Human Relations seeks to accommodate human need for moral solidarity
- Braverman sees liberated workers only under socialism
- Human-Centred School contemplates work reforms under any system

Summary of approaches - 2

"Until recently, employers have concentrated on gaining clerical productivity increases through Taylorization.... The idea seems to have been that if you tidy up work so that certain people only do certain operations there should be efficiency gains"

Smith (1989)

Summary of approaches - 3

"Staff must be permitted to float up to their own levels of ability, rather than be rigidly pre-categorised according to formal educational qualifications and recruitment tiers.... All this points towards a much more imaginative approach to systems design and technical agreements than we have seen to date."

Smith(1989)

Monitoring people at work

- Many studies show that monitoring can lead to a decrease of worker autonomy and an increase in stress, particularly when used for performance evaluation (Long, 1989)
- Others find that "workers appear to be less resentful of errors unambiguously attributable to them than being blamed for those committed by others" (Rothwell, 1984)

Monitoring

- Before technology
 - Done by people
 - Episodic
 - Workers knew when they were watched
 - Limited by sensory capabilities
 - Direct, personal
- After technology
 - Done increasingly by machines
 - Offices as well as factories
 - Workers may not know when they are watched
 - Can be done any time
 - ..from distant locations

Privacy

- Privacy is an essential component of individual autonomy and dignity
- The notion of privacy is changing
- boundaries between what is acceptable and what is not are blurred
- Intrusions that were unacceptable are now commonplace

What can you do to protect it?

- Ask why your company wants to monitor people:
 - what data will be collected
 - how they will be collected
 - how they will be used
 - what feedback will be provided to those who are being observed

What can you do to protect it?...

- You can encourage:
 - visible monitoring
 - users/customers to agree in writing to being monitored
 - that customers agree in writing to use the software only for quality control
- You can refuse to be part of a project that compromises basic human privacy values

Team Management

- Involves the following main activities:
 - Organisational Planning
 - Staff Acquisition
 - Team Development

Organizational Planning - 1

- Inputs:
 - Project Interfaces
 - Staffing Requirements
 - Constraints
- Tools and Techniques:
 - Templates
 - Human Resource Practices
 - Organizational Theory
 - Stakeholder Analysis

Organizational Planning - 2

- Outputs:
 - Role and Responsibility Assignments
 - Staffing Management Plan
 - Organizational Chart
 - Supporting Detail

Staff Acquisition - 1

- Inputs:
 - Staffing Management Plan
 - Staffing Pool Description
 - Recruitment Practices
- Tools and Techniques:
 - Negotiations
 - Pre-assignment
 - Procurement
- Outputs:
 - Staff Assigned
 - Team Directory

Team Development - 1

- Inputs:
 - Allocated Staff
 - Project Plan
 - Management Plan
 - Performance Appraisal Reports/Techniques
 - External Feedback

Team Development - 2

- Tools and Techniques:
 - Team Building Activities
 - General Management Skills
 - Reward and Recognition systems
 - Collation
 - Training

Team Development - 3

- Outputs:
 - Performance Improvements
 - Inputs into Performance Appraisals

Project Interfaces

- They fall into three categories:
 - **Organizational Interfaces** – Among different organizational units (Finance, Sales and Information Systems).
 - **Technical Interfaces** – Formal and informal reporting between teams (eg. Hardware and Software Engineers).
 - **Interpersonal Interfaces** – normally among different individuals in the team.

Staffing Requirements

- What kind of skills are required?
- What type of individuals are required?
- What is the ideal group structure?
- Time frame available for staffing.
- Staff form a subset of overall resource requirements identified during resource planning.

Constraints

- Factors that limit the project team's options:
 - Team structure may be dictated by organizational culture (or procedures).
 - Unions and employee groups impose some constraints.
 - Educational qualifications (may be similar level but from different universities).

Templates

- Using a common template will help in developing the required documentation.
- Templates can be patterns that have worked before (like an organizational structure, documentation structure, reporting structure etc...).

Stakeholder Analysis

- Stakeholders are more often than not part of the overall team:
 - Political issues have to be carefully managed.
 - Needs vs wants of the stakeholders should be carefully understood.
 - Ideally it should be very formal.

Roles/Responsibility Assignment

- Who does what?
- Who decides what?
- A RAM (Responsibility Assignment Matrix is often used, see next slide).
- In large projects RAMs may be developed at various levels.

RAM - Example

| Phase \ People | A | B | C | D | E |
|----------------------|---|---|---|---|---|
| Requirements | S | R | A | P | P |
| Analysis | S | | A | | |
| Design | S | | A | I | |
| Implementation/Build | | R | S | | S |

Legend: P = Participant, A = Accountable, R = Review Required, I = Input Required, S = Sign-Off Required

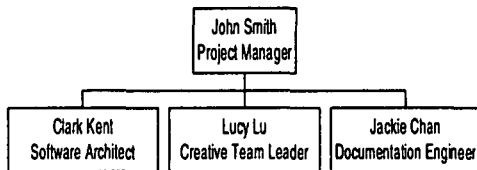
Staffing Management Plan

- Describe when and how human resources will be brought onto and taken off the teams in a project.
- Often include resource histograms (Resource/Skill usage over Months/Weeks) depending on the measurement/tracking granularity.

Organization Chart

- A graphical display of project reporting relationships (with Names and Roles).
- Should be formal (on smaller projects this may be communicated informally, but it achieves best results when formally presented).
- A specialised role and responsibility detail should be made available.

Organization Chart- Example



Software Project Management

49

Organization Chart-Limitations

- The main problem is that it does not capture and convey all the dimensions in which a team works.
- Only the reporting, problem escalation chain is represented.
- In practice seniority, skill and depth of knowledge create people of higher influence on the project. But they may be shown lower on the organisational chart.

Software Project Management

50

References - 1

- Forester, T. (ed) (1989) *Computers in the human context: Information technology, productivity and people*, Basil Blackwell.
- Franke, R. H., Technological Revolution and productivity decline: The case of US banks.
- Smith, S., Information technology in bank: Taylorization or human centred systems?
- Marx, G. T., and Sherizen, S., Monitoring on the job.
- PMI Standards Committee (1996) *A Guide to the Project Management Body of Knowledge*.

Software Project Management

51

References - 2

- Grudin, J. (1993) "Obstacles to participatory design in large product development organizations", in D. Schuler and A. Namioks (ed.), *Participatory Design: Principles and Practices*, Erlbaum.
- Lindgaard, G. (1995) "Cementing human factors into product design: Moving beyond policies", *Proceedings iHFT'95 International Human Factors in Telecommunications*, Melbourne, 1995.
- Cougar, J.D., and Zwacki, R.A. (1980) *Managing and Motivating Computer Personnel*, Addison-Wesley.

Software Project Management

52

References - 3

- Hackman, J. R., and Oldham, G. R. (1980) *Work Redesign*, Addison-Wesley.
- Thomsett, R. (1993) *Third Wave Project Management*, Prentice-Hall.
- Thomsett, R. (1990) "Building Effective Project Teams", *American Programmer*, Summer 1990.
- Vroom, V. H., and Deci, E. I. (1978) *Management and Motivation*, Penguin.

Software Project Management

53