

An Introduction to Project Management – The changing landscape of Software development

- Sources: 1. B. Bruegge and A. H. Dutoit, Object-Oriented Software Engineering: Using UML, Patterns, and Java (Chapter 14)
2. Roger S. Pressman, Software Engineering - a Practitioner's Approach, 5th edition, ISBN 0073655783, 2001, McGraw-Hill
3. Robert K. Wysocki, Effective Software Project Management, ISBN-13: 9780764596360 ISBN-10: 0764596365, 2006 Wiley

What do you learn in this course?

- ❖ Better understand the Software Lifecycle
 - ♦ Learn about different software lifecycles
 - ♦ Greenfield Engineering, Interface Engineering, Reengineering
- ❖ Learn about the differences between Process vs Product
- ❖ Learn more about basic project management activities and dealing with resources:
 - ♦ Schedule: How to map activities into time
 - ♦ Organization and People: How to set up a project organization
 - ♦ Cost: How to estimate the cost of a project
- ❖ Learn how to deal with change and uncertainty
- ❖ Learn how to set up a project plan, how to control its execution
- ❖ Learn how to become a leader, how to deal with teams, and make decisions in the context of project management trade-offs
 - ♦ Cost vs People vs Schedule, Buy vs Build, ...

Objectives of the Class

- ❖ Appreciate Software Engineering management
 - ♦ Manage the construction of complex software systems in the context of frequent change
- ❖ Understand how to
 - ♦ produce a high quality software system within time
 - ♦ while dealing with complexity and change
- ❖ Appreciate that managerial knowledge is needed when developing software system
- ❖ We assume that you have the technical knowledge (e.g. SYSC 3100 and SYSC 4800)

Software Production has a Poor Track Record Example: Space Shuttle Software

- ❖ Cost: \$10 Billion, millions of dollars more than planned
- ❖ Time: 3 years late
- ❖ Quality: First launch of Columbia was cancelled because of a synchronization problem with the Shuttle's 5 onboard computers.
 - ♦ Error was traced back to a change made 2 years earlier when a programmer changed a delay factor in an interrupt handler from 50 to 80 milliseconds.
 - ♦ The likelihood of the error was small enough, that the error caused no harm during thousands of hours of testing.
- ❖ Substantial errors still exist.
 - ♦ Astronauts are supplied with a book of known software problems "Program Notes and Waivers".

Software Engineering: A Problem Solving Activity

- ❖ **Analysis:** Understand the nature of the problem and break the problem into pieces
- ❖ **Synthesis:** Put the pieces together into a large structure
- ❖ For problem solving we use
 - ♦ Techniques(Methods): **Formal procedures for producing results using some well-defined notation**
 - ♦ Tools: **Instrument or automated systems to accomplish a technique**
 - ♦ Methodologies: **Collection of techniques applied across software development and unified by a philosophical approach**
- ❖ **Where does Management come in?** When the available resources to solve the problem are limited (time, people, budget), or if we allow the problem to change.

Software Engineering: Definition

Software Engineering is a collection of techniques, methodologies and tools that help with the development of

a high quality software system

with a given budget

before a given deadline

while change occurs.

What is Software?

- ❖ Software is engineered – it is not manufactured in the classical sense. Software costs are concentrated in engineering.
- ❖ Software doesn't wear out – Figure 1.1 depicts failure rate as a function of time for hardware. Unlike hardware, software is not susceptible to maladies that causes hardware to wear out. In theory, the failure rate for software should take the form of figure 1.2
- ❖ Software is complex
- ❖ Software is a 'differentiator'
- ❖ Software is like an 'aging factory'

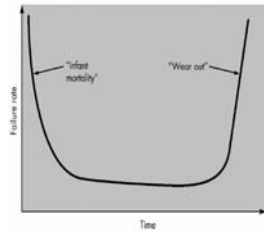
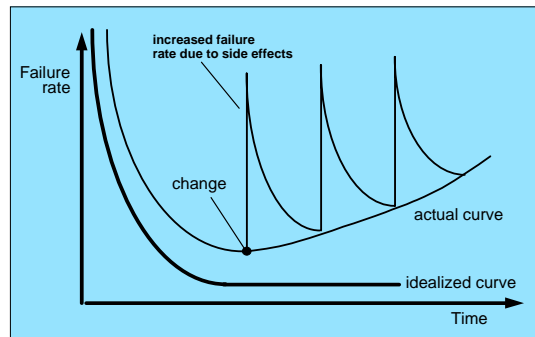
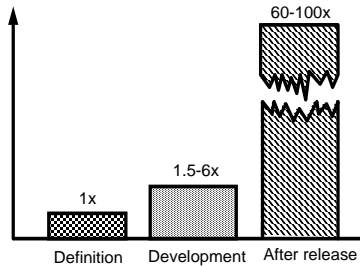


Figure 1.1 – Failure curve for hardware

Wear vs. Deterioration



The Cost of Change



Software Poses Challenges

- ❑ How do we ensure the quality of the software that we produce?
- ❑ How do we meet growing demand and still maintain budget control?
- ❑ How do we upgrade an aging "software plant?"
- ❑ How do we avoid disastrous time delays?
- ❑ How do we successfully institute new software technologies?

A solution is to have effective Software Development Project Management

What is a Software Development Project?

- ❖ A software development project is a complex undertaking by two or more persons within the boundaries of **time**, **budget**, and **staff resources** that produces new or enhanced computer code that adds significant business value to a new or existing business process.
- ♦ **Although this is a restrictive definition, it does apply to the generality of software development projects even industrial processes.**

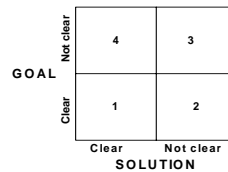
What is Software Development Project Management?

- ❖ Software development project management (SDPM) is the discipline of **assessing the characteristics of the software to be developed**, **choosing the best fit software development life cycle**, and then **choosing the appropriate project management approach** to ensure *meeting the customer needs for delivering business values as effectively and efficiently as possible*.
- ❖ A SDPM strategy is an integration of a *software development life cycle* and a *project management life cycle* into a customer-facing approach that will produce maximum business value regardless of the obstacles that may arise.
 - ♦ **SDPM is an emerging discipline. Although the two components that define it are not new but what is new is the integration of those components to produce an effective SDPM environment.**

SDPM Strategy

- ❖ Project managers would like to what constitutes software development project management and how to do it.
- ❖ Three other questions that are more operationally focused are:
 - What are the characteristics of the software to be developed?
 - What software development approach is appropriate for building the software?
 - What project management approach is appropriate for managing the chosen software development process?
- ❖ These questions are meant to be answered in the order listed. Example, given that both the software approach and the project management approach are fixed (i.e. linear). Do you operate like a solution out looking for a problem? Wouldn't you rather let the characteristics of the problem drive your choices for solution approach? Hmm!

What are the Characteristics of the Software to be Developed?



- Software development landscape is like a two dimensional grid like the one shown above

- ❖ **Quadrant 1: Goal and Solution are clearly specified.** This is the best possible worlds, but it is also the least likely to occur in today's fast-paced, continuously changing business world. Software projects in this quadrant are familiar to the organization and the developers. Perhaps similar projects have been done several times before.
 - Low complexity
 - Well-understood technology infrastructure
 - Low risk
 - Experienced and skilled developer teams

- ❖ Quadrant 2 – Goal is clearly specified but solution is not
 - Close to 70% of software projects fall into quadrant 2.
 - Quadrant 2 projects present a different challenge and need a different approach compare to quadrant 1 projects. The approach must be driven by the characteristics of the project.
- ❖ Quadrant 3 – Goal and solution are not clearly specified
 - These projects have no clear goal and solution. With some planning, the project can proceed through several iterations until it converges on an acceptable goal and solution. If not the customer might pull the plug from the project and look for alternative approaches.
- ❖ Quadrant 4 – Goal is not clearly specified but the solution is
 - This represents projects whose goals are not known but whose solutions are. This is an impossible situation. It would be equivalent to solutions out looking for problems.
 - Well, we have seen professional services organizations that practice such approaches. They advocate a one-size-fits all approach, which has never been successful.

What software development approach is appropriate for building the software?

- ❖ Quadrant 1 – Because all the information about the development project is known and is considered stable, the appropriate development model is the one that gets to the end as quickly as possible.
- ❖ Quadrant 2 – an approach that includes as much of the solution as is known at the time should work quite well. That approach should allow for the customer to examine, in the sense of a prototype what is in the solution approach.
- ❖ Quadrant 3 – If goal clarity is not possible at the beginning of the project, the situation is much like a pure research and development project. In this case you use an approach that clarifies the goal and contributes to the solution at the same time.
- ❖ Quadrant 4 – Okay! You have the solution; now all you need is to find the problem. Yes, this is the stuff that academic articles are often made of. Well, post your solution and hope someone finds the problem that matches it. E.g. 3M Post-it Note saga.

What project management approach is appropriate for managing the chosen software development process?

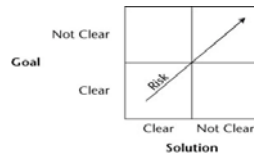
- ❖ As you move through the quadrants from clarity to lack of clarity, the project management processes you use must track with the needs of the project.
- ❖ Remember that “Lots is bad, less is better, and least is best.” i.e. do not burden the project manager and the team with needless planning and documentation that will just hinder their efforts. Too much structure stifles creativity. Too little structure breeds inefficiency.
- ❖ Quadrant 1 projects are plan-driven, process-heavy, and documentation-heavy.
- ❖ As you move to quadrant 2 and 3 projects, heaviness gives way to lightness. Plan-driven gives way to value-driven, rigid process gives way to adaptive process, and documentation is largely replaced by tacit knowledge that is shared among approaches that fall in the agile project management taxonomy.

Complexity/Uncertainty Domain of SDPM

- ❖ Complexity and uncertainty are positively correlated with one another. As software development projects become more complex, they become more uncertain. This follows from at least four factors:
 - **Requirements** – As project complexity increases, the likelihood of nailing requirements decreases. In a complex software product the extent of the number of requirements, functionality, and features can be staggering. Some will conflict, some will be redundant, and some will be missing.
 - **Flexibility** – as project complexity increases, so does the need for process flexibility. Increased complexity brings with the need to be creative and adaptive. This is difficult to achieve in the company of rigid processes.
 - **Adaptability** – this is directly related to the extent to which the team members are empowered to act. When a project is less certain in terms of requirements, functionality, and features, the more the need to be adaptable with respect to process and procedure.
 - **Change** – As complexity increases, the frequency and need to receive and process change requests increase as well. A plan driven software development project is not designed to effectively respond to change.

Risk versus the complexity/uncertainty domain

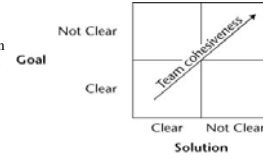
- ❖ Risk increases as you move from quadrant 1 to 2 to 3.
- ❖ In quadrant 1 you know the goal and the solution and can build a definitive plan for getting there. The focus can shift to process failure.
- ❖ Because a list of candidates risks have been compiled over past similar projects, the likelihood, impact, and the appropriate mitigations is known and documented.
- ❖ In quadrant 2 two forces come into play – the SDPM strategy becomes more flexible and lighter; and at the same time, the product risk increases.



- In quadrant 3, risk is the highest because you are in a research and development environment. Process risk is almost nonexistent because the ultimate in flexibility has been reached but the product risk is extremely high.

Team cohesiveness versus the complexity/uncertainty domain

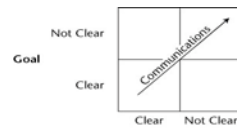
- ❖ In quadrant 1 the successful team doesn't really have to be a team at all. A team of specialists will do. The plan is sacred and the plan guides the team through their tasks.
- ❖ The situation quickly changes in quadrant 2 or 3 project. There is gradual shift from a team of specialists to team of generalists.
- ❖ Teams become self-sufficient and self-directing as the project moves from a quadrant 2 to a quadrant 3.
- ❖ Quadrant 1 team are not co-located while quadrant 2 and 3 teams are co-located.



- Research has shown that co-location adds significantly to the successful completion of the project. The figure above reflects this from a loosely formed team to one that is tightly coupled.

Communications versus the complexity/uncertainty domain

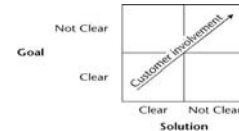
- ❖ Lack of timely and clear people-to-people communications has been shown to be the single most frequent reason for project failure.
- ❖ As you move in the direction of increased complexity and heightened uncertainty, communication requirements increase and change.
- ❖ When complexity and uncertainty are low, the predominant form of communications is written.
- ❖ As uncertainty and complexity increases, written communication give way to verbal communication.



- With increased uncertainty and complexity, the burden of plan-driven approaches is lightened, and the communications requirements of value-driven approaches take over.
- Value-driven communications approaches are the derivatives of meaningful customer involvement.

Customer involvement versus the complexity/uncertainty domain

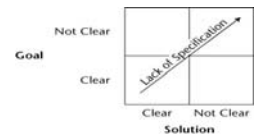
- ❖ Customer involvement is usually limited in quadrant 1 projects because they are team-driven.
- ❖ The customers take an important role in quadrant 2 and 3 projects.
- ❖ Meaningful customer involvement can be daunting task because of the following three reasons:
 - **The customer's comfort zone** - traditionally customers have been trained to be passive but, that is changing. Now you have a lot of technically sound customers that know what it takes to build a software.
 - **Ownership by the customer** - this is critical but care need to be taken so that the customer will not delay the project.



- **Customer sign-off** - This is often an anxiety-filled task that you ever ask of your customer because customers think that they are signing their lives away when they approve a document or a deliverable. Your task is to dispel that perception. This is always a challenge.

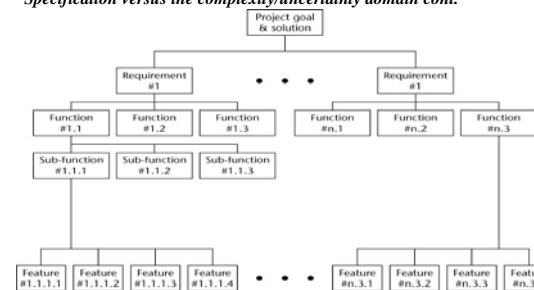
Specification versus the complexity/uncertainty domain

- ❖ What does this means? Simply put, it advises you that the choice of SDPM strategy should be based on an understanding of the confidence you have that the specifications have been completely and clearly defined and documented.
- ❖ Also, that scope change requests will not arise from any shortcomings in the specifications documents.
- ❖ As that specification certainty diminishes, your best choices lie in the iterative strategies that populate quadrant 2 projects.



- Finally, if you are not sure that you have clearly and completely documented the specifications, then your SDPM strategy takes on the flavour of the research and development strategies that populate quadrant 3.

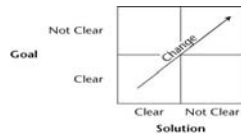
Specification versus the complexity/uncertainty domain cont.



- ❖ Uncertainty at the requirements level has more impact on choice of software development approach than does uncertainty at the functionality level, which has more impact than that at the features level.

Change versus the complexity/uncertainty domain

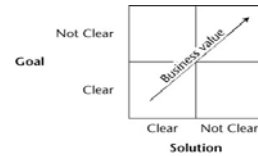
- ❖ The less you know about requirements, functionality, and features, the more you have to expect change.
- ❖ Quadrant 1 – you know everything (almost) that needs to be known so change is minimal.
- ❖ Quadrant 2 – Any change in this quadrant comes about through the normal learning process that takes place in any software development project.
- ❖ Quadrant 3 – The projects in this quadrant require change in order to have any chance at finding a successful solution. Change is the only vehicle that will lead to solution.



The figure above reflects the frequency of change as projects move across the landscape.

Business value versus the complexity/uncertainty domain

- ❖ It seems all software projects are created to return business value to the enterprise. This is all true. However, traditional project approaches focus on meeting the plan-driven parameters: time, cost, scope. When originally proposed the business climate was such that the proposed solution was the best that could be had. Unfortunately business world is not static.
- ❖ Quadrant 1 development projects are not equipped with the right stuff to deliver business value.
- ❖ As you move from quadrant 1 to 2 to 3, risk increases and that means that higher-values projects need to be commissioned because the



expected business value of a project is the product of (1 – risk) and value. Risk here is the probability of failure and the probability of success is therefore (1 – risk).

- Simply put, whatever SDPM strategy you adopt for the project, it must be one that allows redirection as business conditions change. More uncertainty means more redirection.

Balancing Staff, Process, and Technology

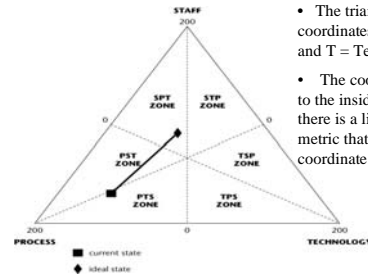
- ❖ Staff i.e. the skill and competency profile of the project team drives the choice of software development process and project management process to be employed, and together staff and process drive the choice of technology.
- ❖ These three factors together form the SDPM strategy.



- ❖ The balance is achieved by:
 - Assessing the available staff resources as compared to the skill and competencies needed for the project.
 - The chosen project team then determines the SDPM strategy that best meets the needs of the project and aligns with the team's capacity to deliver.
 - Using the chosen strategy, the team will now select the technology infrastructure that best supports the team's capacity to deliver.

Balancing Staff, Process, and Technology cont.

- ❖ The balance achieved by the above choices is represented by the triangle below:



- The triangle shows three coordinates (S = staff, P = process, and T = Technology).
- The coordinates are constrained to the inside of the triangle because there is a linear constraint on the metric that measures each coordinate.

Balancing Staff, Process, and Technology cont..

- ❖ In this example, the sum total of the assessed values of each coordinate is 200.
- ❖ Also, the ordering of the three letters (S, P, T) is meaningful.
- ❖ The proximity of each vertex to the data point determines the ordering.
 - E.g., in the figure above, the current state is closest to the Process vertex, next closest to the Technology vertex, and furthest from Staff vertex. That results in the labeling PTS.
 - All of the data points that have that property fall in the zone labeled PTS.
- ❖ Knowing the current state and ideal, or desired, end state, you can develop a plan that will migrate the organization to its desired end state. The figure below shows some of the migration strategies.

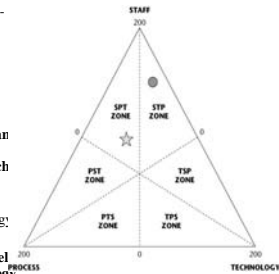
From Zone	To Zone	Comments on the Transition
TPT	TSP	In this situation technology may have constrained the formation of project management processes because there was little or no input from staff. To correct this situation staff could be empowered to improve project management processes. This may result in some reversals or prior technology decisions.
TSP	STP	Even though technology might be a constraint, staff has had an opportunity to define project management processes. Staff needs to be empowered to make decisions regarding the appropriate technology as it relates to project management processes.
STP	SPT	In this situation the staff are in a enviable position. The remaining task is to create a more balanced relationship between process and technology. That will require slow changes so that the technology environment is adapted to provide better support for project management processes.
PTS	PST	There may be good coherence between project management processes and the technology to support it but it would have happened without much priority given to the role of staff. That situation can begin to change by commissioning the staff to work on technology improvement initiatives. This may result in reversing prior decisions.
PST	SPT	This is a strong starting position. Project management processes are a high priority for the organization. Technology has been implemented to support both process and staff. The remaining step is to move staff into a higher priority position for further enhancement of the project management environment and the technical support of it.

Balancing Staff, Process, and Technology cont...

- ❖ Note that the migrations are between neighboring zones only.
- ❖ E.g. if the current state is zone TPS and the desired zone is SPT, then the migrations would be from TPS to TSP to STP, and to SPT.

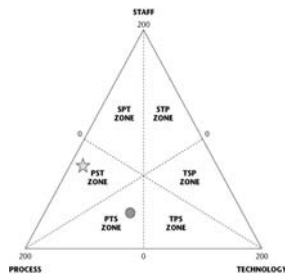
Staff-Driven Environments

- ❖ The figure below illustrates two people-driven environments.
- 1. Denoted by star: Staff drives SPDM strategy, and together S and P drive technology.
 - This model shows that you are leveraging the skill and competency capacity of your team and the characteristics of the project to decide how to approach the project from an SDPM perspective.
- 2. Denoted by circle: Staff drive technology and together S and T drive process.
 - The only problem with this model is that the choice of SPDM strategy will be constrained by the earlier decision on technology.



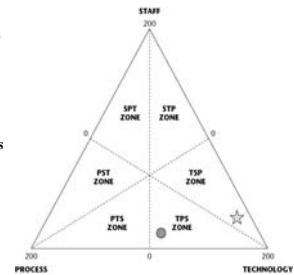
Process-Driven Environments

- ❖ The figure below shows two process-driven environments.
- 1. Denoted by star: SDPM strategies derives staff, and together P and S drive choice of technology infrastructure.
 - This represent an organization with only one SDPM strategy – one size fits all. This is not good for iterative, adaptive, and extreme world.
- 2. Denoted by circle: SDPM strategy drives the choice of technology, and together P and T drive the choice of staff.
 - This means that staff need to be put in place in timely and effective manner. Well, this might or might not be a problem.

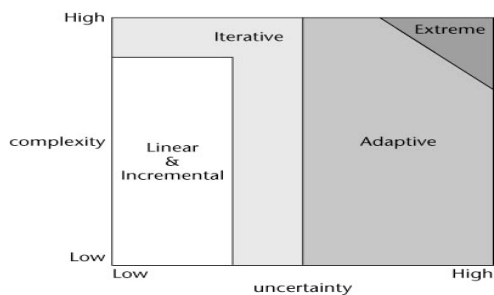


Technology-Driven Environments

- ❖ Two technology-driven environments are shown below:
- 1. Denoted by star: Technology infrastructure drives the choice of staff, and together T and S drive the choices for SPDM strategy.
 - This situation is workable if the technology infrastructure is not a binding constraint on good project performance.
- 2. Denoted by circle: Technology infrastructure drives the choice of SDPM strategy, and together T and P drive the choice of staff.
 - This can work if the staff is given the authority to adapt the SDPM strategy to the project situation.



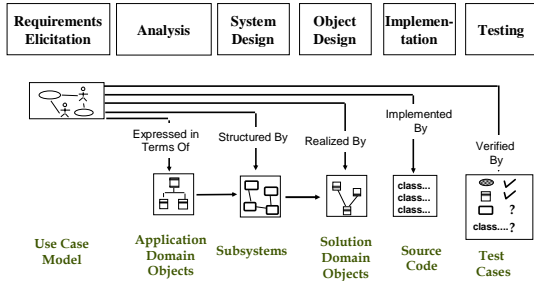
A glimpse of Contemporary Software Development Landscape



Software Lifecycle

- ❖ Software lifecycle:
 - Set of activities and their relationships to each other to support the development of a software system
- ❖ Typical lifecycle questions:
 - Which activities should I select for the software project?
 - What are the dependencies between activities?
 - Can I break these activities into more manageable parts?
 - How should I schedule the activities?
 - How do I assign people to these activities?
 - How do I control the execution of the activities?
 - What do I do if the activities are not proceeding as planned?

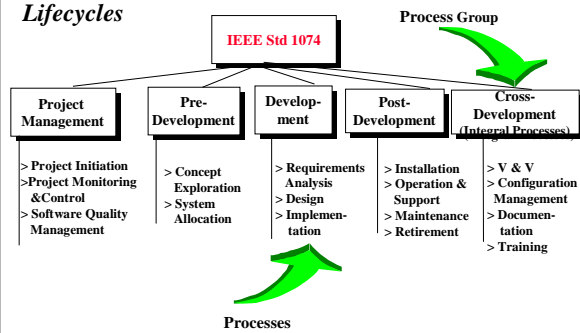
Example of Software Lifecycle Activities



Standard for modeling lifecycles IEEE Std 1074

- ❖ The IEEE Std 1074 defines
 - ◆ The set of activities that constitute mandatory processes for the development and maintenance of software
 - ◆ Management and support processes through the entire lifecycle
 - ◆ From concept exploration through retirement
- ❖ It does not define a software lifecycle on its own
 - ◆ This must be done by the project manager
 - ◆ Also called ``Tailoring the software lifecycle``

IEEE Std 1074: Standard for Software Lifecycles



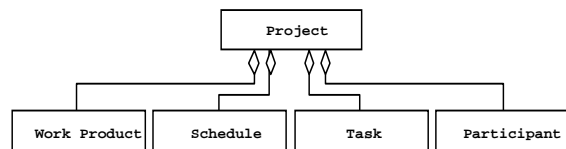
Basic Definitions: Project and Project Plan

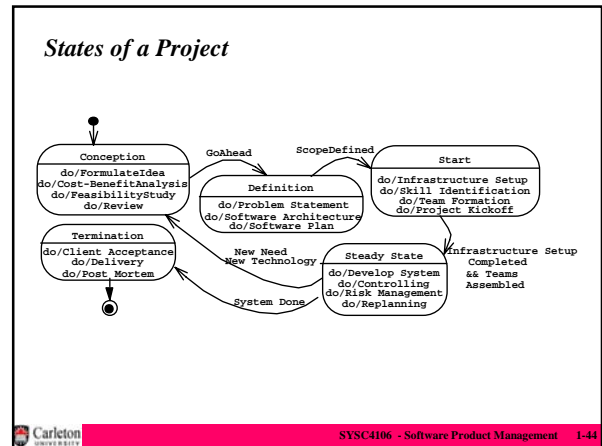
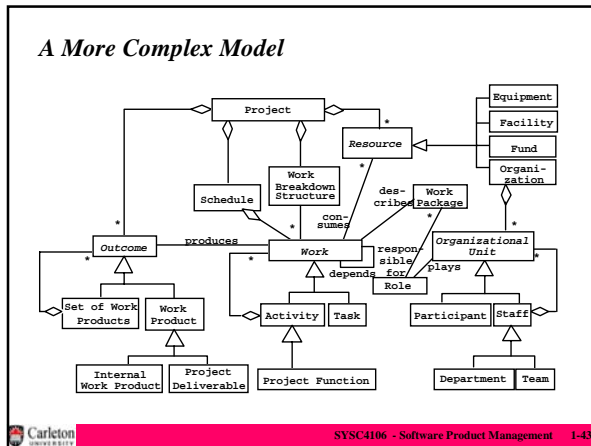
- ❖ Software Project:
 - ◆ All technical and managerial activities required to deliver the deliverables to the client.
 - ◆ A software project has a specific duration, consumes resources and produces work products.
 - ◆ Management categories to complete a software project:
 - ◆ Tasks, Activities, Functions
- ❖ Software Project Management Plan:
 - ◆ The controlling document for a software project.
 - ◆ Specifies the technical and managerial approaches to develop the software product.
 - ◆ Companion document to requirements analysis document:
 - ◆ Changes in either document may imply changes in the other document.
 - ◆ The SPMP may be part of the project agreement.

Definitions

- ❖ **Teams:** represent sets of participants who work on a common problem
- ❖ **Roles:** represent sets of responsibilities. Roles are used to distribute work to participants within a team
- ❖ **Work products:** represent the deliverables and intermediate products of the project. Work products are the visible results of the work
- ❖ **Tasks:** represent work in terms of sequential steps necessary to generate one or more work products
- ❖ **Schedules:** represent the mapping of a task model onto a time line. A schedule represents a work in terms of calendar time.
- ❖ The project manager and the team leaders construct and maintain these model elements throughout development.

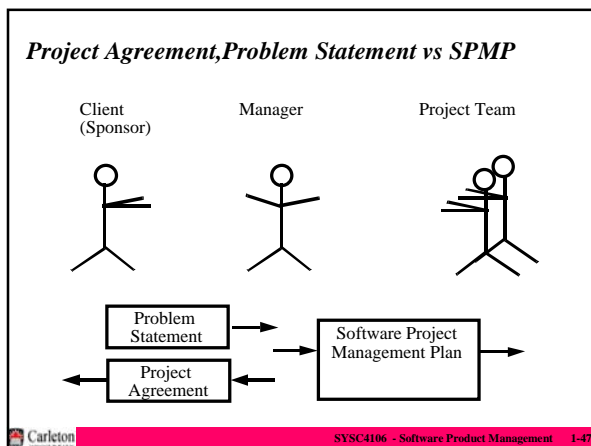
Components of a Project





- ### Project Agreement
- ❖ Document written for a client that defines:
 - ♦ the scope, duration, cost and deliverables for the project.
 - ♦ the exact items, quantities, delivery dates, delivery location.
 - ❖ Client: Individual or organization that specifies the requirements and accepts the project deliverables.
 - ❖ Can be a contract, a statement of work, a business plan, or a project charter.
 - ❖ Deliverables (= Work Products that will be delivered to the client):
 - ♦ Documents
 - ♦ Demonstrations of function
 - ♦ Demonstration of nonfunctional requirements
 - ♦ Demonstrations of subsystems
- Carleton SYSC4106 - Software Product Management 1-45

- ### IEEE Std 1058: Standard for Software Project Management Plans (SPMP)
- ❖ What it does:
 - ♦ Specifies the format and contents of software project management plans.
 - ♦ It provides a standard set of abstractions for a project manager or a whole organization to build its set of practices and procedures for developing software project management plans
 - ♦ Abstractions: Project, Function, Activities, Tasks
 - ❖ What it does not do:
 - ♦ It does not specify the procedures or techniques to be used in the development of the plan
 - ♦ It does not provide examples .
- Carleton SYSC4106 - Software Product Management 1-46



- ### Software Project Management Plan Template
- ❖ 0. Front Matter
 - ❖ 1. Introduction
 - ❖ 2. Project Organization
 - ❖ 3. Managerial Process
 - ❖ 4. Technical Process
 - ❖ 5. Work Elements, Schedule, Budget
 - ❖ Optional Inclusions
- Carleton SYSC4106 - Software Product Management 1-48

SPMP Part 0: Front Matter

- ❖ Title Page
- ❖ Revision sheet (update history)
- ❖ Preface: Scope and purpose
- ❖ Tables of contents, figures, tables

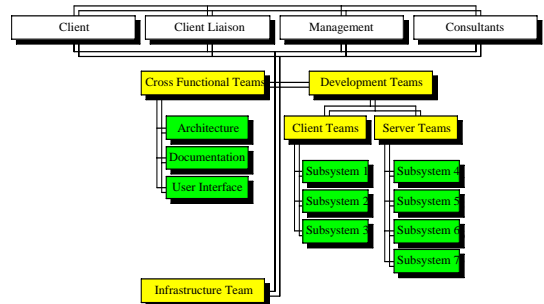
SPMP Part 1: Introduction

- ❖ 1.1 Project Overview
 - ♦ **Executive summary: description of project, product summary**
- ❖ 1.2 Project Deliverables
 - ♦ **All items to be delivered, including delivery dates and location**
- ❖ 1.3 Evolution of the SPMP
 - ♦ **Plans for anticipated and unanticipated change**
- ❖ 1.4 Reference Materials
 - ♦ **Complete list of materials referenced in SPMP**
- ❖ 1.5 Definitions and Acronyms

SPMP Part 2: Project Organization

- ❖ 2.1 Process Model
 - ♦ **Relationships among project elements**
- ❖ 2.2 Organizational Structure
 - ♦ **Internal management, organization chart**
- ❖ 2.3 Organizational Interfaces
 - ♦ **Relations with other entities (subcontractors, commercial software)**
- ❖ 2.4 Project Responsibilities
 - ♦ **Major functions and activities; nature of each; who's in charge**
 - ♦ **Matrix of project functions/activities vs responsible individuals**

Organizational Structure Example



SPMP Part 3: Managerial Process

- ❖ 3.1 Management Objectives and Priorities
 - ♦ **Describes management philosophy, priorities among requirements, schedule and budget**
- ❖ 3.2 Assumptions, Dependencies and Constraints
 - ♦ **External events the project depends on, constraints under which the project is to be conducted**
- ❖ 3.3 Risk Management
 - ♦ **Identification and assessment of risk factors, mechanism for tracking risks, implementation of contingency plans**
- ❖ 3.5 Monitoring and Controlling Mechanisms
 - ♦ **Frequency and mechanisms for reporting**
- ❖ 3.4 Staffing Plan
 - ♦ **Numbers and types of personnel required to conduct the project**

Examples of Risk Factors

- ❖ Contractual risks
 - ♦ **What do you do if the client becomes bankrupt?**
- ❖ Size of the project
 - ♦ **What do you do if you feel the project is too large?**
- ❖ Complexity of the project
 - ♦ **What do you do if the requirements are multiplying during analysis? („requirements creep“)**
- ❖ Personal
 - ♦ **How do you hire people? Is there a danger of people leaving the project?**
- ❖ Client acceptance
 - ♦ **What do you do, if the client does not like the developed prototype?**

SPMP Part 4: Technical Process

- ❖ 2.1 Methods, Tools and Techniques
 - ♦ Specify the methods, tools and techniques to be used on the project
- ❖ 2.2 Software Documentation
 - ♦ Describe the documentation plan
- ❖ 2.3 Project Support Functions
 - ♦ Plans for (at least) the following project support functions.
 - ♦ Plan to ensure quality assurance
 - ♦ Configuration management plan (IEEE Std 1042)
 - ♦ Verification and validation plan
 - ♦ The plans can be included in this section or there is a reference to a separate document

SPMP Part 5: Description of Work Packages

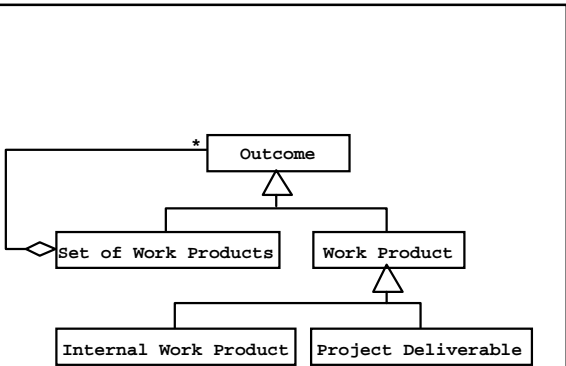
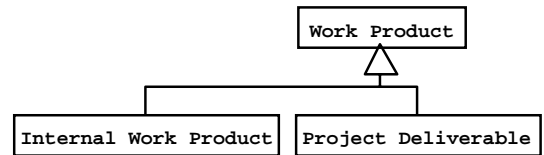
- ❖ Work Breakdown Structure (WBS) (Section 5.1)
 - ♦ Hierarchical decomposition of the project into activities and tasks
- ❖ Dependencies between tasks (Section 5.2)
 - ♦ An important temporal relation preceded by
 - ♦ Dependency graphs showing temporal dependencies of the activities

Work package vs Work product

- ❖ Definitions from the IEEE Standard
- ❖ Work Package:
 - ♦ A specification for the work to be accomplished in an activity or task
- ❖ Work Product:
 - ♦ Any tangible item that results from a project function, activity or task.
- ❖ Project Baseline:
 - ♦ A work product that has been formally reviewed and agreed upon.
 - ♦ A project baseline can only be changed through a formal change procedure
- ❖ Project Deliverable:
 - ♦ A work product to be delivered to the client

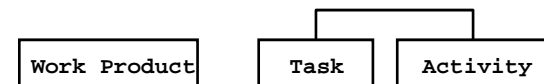
How can we model a work product?

- ❖ What is a tangible item?
 - ♦ Requirements Analysis Document
 - ♦ Software Project Management Plan
 - ♦ Agenda, Minutes
 - ♦ How do we model Tangible Items?



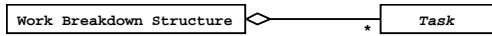
Work products are related to Activities

- ❖ How do we model this?

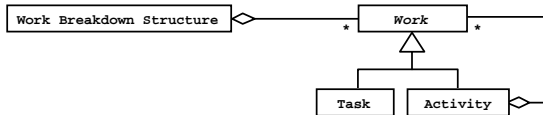


Work Breakdown Structure

- ❖ The hierarchical representation of all the tasks in a project is called the work breakdown structure (WBS). First Version of a UML Model



- ❖ But Tasks are Parts of Activities. What would be a better model?



Creating Work Breakdown Structures

- ❖ Two major philosophies
 - Activity-oriented decomposition ("Functional decomposition")
 - ◆ Write the book
 - ◆ Get it reviewed
 - ◆ Do the suggested changes
 - ◆ Get it published
 - Result-oriented ("Object-oriented decomposition")
 - ◆ Chapter 1
 - ◆ Chapter 2
 - ◆ Chapter 3
- ❖ Which one is best for managing? Depends on project type:
 - Development of a prototype
 - Development of a product
 - Project team consist of many inexperienced beginners
 - Project team has many experienced developers

Estimates for establishing WBS

- ❖ Establishing an WBS in terms of percentage of total effort:
 - ◆ Small project (7 person-month): at least 7% or 0.5 PM
 - ◆ Medium project (300 person-month): at least 1% or 3 PMs
 - ◆ Large project (7000 person-month): at least 0.2 % or 15 PMs
 - ◆ (From Barry Boehm, Software Economics)

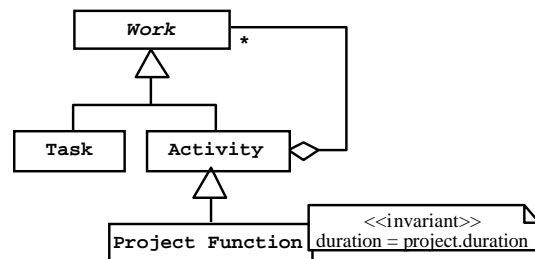
From the WBS to the Dependency Graph

- ❖ The work breakdown structure does not show any temporal dependence among the activities/tasks
 - ◆ Can we excavate before getting the permit?
 - ◆ How much time does the whole project need if I know the individual times?
 - ◆ What can be done in parallel?
 - ◆ Are there any critical activities, that can slow down the project significantly?
- ❖ Temporal dependencies are shown in the dependency graph
 - ◆ Nodes are activities
 - ◆ Lines represent temporal dependencies

Goals of PERT Charts

- ❖ Determination of total project time ("project duration")
- ❖ Determination of the critical path
- ❖ Determination of slack times

UML Model of Tasks, Activities and Project Functions



“Laws” of Project Management

- ❖ Projects progress quickly until they are 90% complete. Then they remain at 90% complete forever.
- ❖ When things are going well, something will go wrong.
- ❖ When things just can't get worse, they will.
- ❖ When things appear to be going better, you have overlooked something.
- ❖ If project content is allowed to change freely, the rate of change will exceed the rate of progress.
- ❖ Project teams detest progress reporting because it manifests their lack of progress.

Summary

- ❖ Software engineering is a problem solving activity
 - ♦ **Developing quality software for a complex problem within a limited time while things are changing**
- ❖ The system models addresses the technical aspects:
 - ♦ **Object model, functional model, dynamic model**
- ❖ Other models address the management aspects
 - ♦ **WBS, Schedule are examples**
 - ♦ **Task models, Issue models, Cost models**
- ❖ Introduction of some technical terms
 - ♦ **Project, Activity, Function, Task, WBS**