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

Sources:

1. Bob Hughes and Mike Cotterell, Software Project management, 5th edition, ISBN 9780077122 , 2009, McGraw Hill – **(Recommended)**
2. Robert D. Austin, Richard L. Nolan, and Shannon O'Donell, The Adventure of an IT Leader, Harvard Business Press, Boston, Massachusetts, 2009, ISBN 978-1-4221-4660-6 (**Recommended**). You are encouraged to get a copy of this book. It is NOT expensive
3. Robert T. Futrell et al., Quality Software Project Management, Software Quality Institute Series, 2002 Prentice Hall PTR, ISBN 0-13-091297-2
4. Bernd Bruegge and Allen H. Dutoit, Object-Oriented Software Engineering: Using UML, Patterns, and Java, 2nd edition, ISBN 0-13-0471100, Prentice-Hall, 2004
5. Roger S. Pressman, Software Engineering - a Practitioner's Approach, 7th edition, ISBN 9780071267823, 2010, McGraw-Hill

What do you learn in this course?

- Better understanding of the Software Lifecycle
  - **Learn about different software lifecycles**
- 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
- I assume that you have the technical knowledge

Introduction: Why is project management important?

- Large amounts of money are spent on ICT e.g. IT spending in the UK in 2017 is roughly $124.8 billions and only $4.27 billions on road building
- Project often fail – Standish Group claim only a third of ICT projects are successful. 82% were late and 43% exceeded their budget.
- The Canadian "Phoenix system" is a recent example of how not manage a project especially software project.
- Poor project management a major factor in these failures
- https://www.ic.gc.ca/eic/site/ict-tic.nsf/eng/h_it07229.html

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? (Wear vs. Deterioration)

- 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, but is delivered "broken."

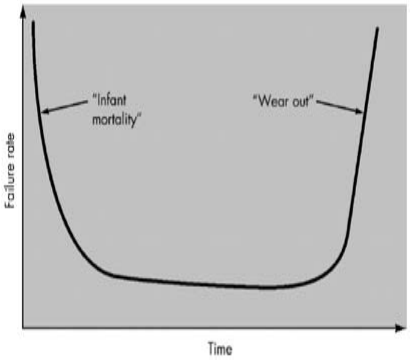
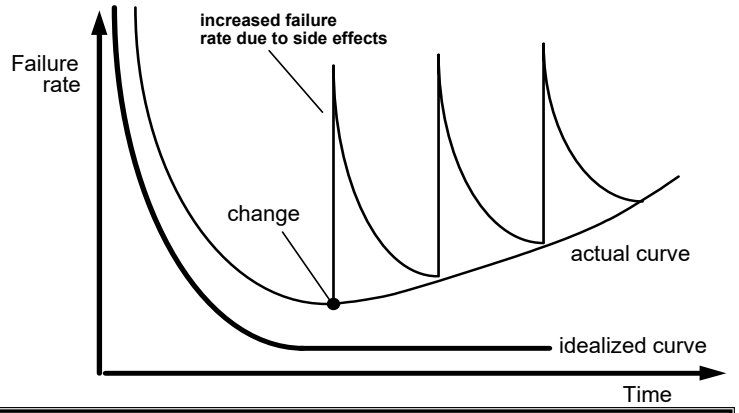


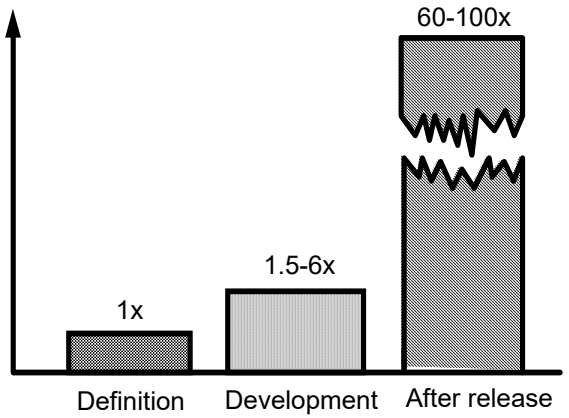
Figure 1.1 – Failure curve for hardware

What is Software? Wear vs. Deterioration cont.



- Theoretically, the failure rate for software resemble the figure above
- Software is complex
- Software is a 'differentiator'
- Software is like an 'aging factory'

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 Project Management

What is a Project? Operation? And Stakeholders?

- **Projects** are unique; temporary in nature and have definite start dates and definite end dates.
- The project is completed when its goals and objectives are accomplished to the satisfaction of the stakeholders.
- Projects exist to bring about a product, service, or result that did not exist before.
- A project is successful when it achieves and meets or exceeds the expectations of the stakeholders.
- **Operations** (vs. projects) are ongoing and repetitive. They involve work that is continuous without an ending date, and often repeat the same process and produce the same results.
- At the completion of a project, the end product may get turned over to organization's operational areas for ongoing maintenance.
- **Stakeholders** are those folks (or organizations) with a vested interest in the project.

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
  1. **assessing the characteristics of the software to be developed,**
  2. **choosing the best fit software development life cycle, and**
  3. **choosing the appropriate project management approach**
- to ensure *meeting the customer needs for delivering business values as effectively and efficiently* as possible.

What is Software Development Project Management? Cont.

- A SDPM strategy is an integration of a
  1. ***software development life cycle* and**
  2. **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 know 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!

Are software projects really different from other projects?

Not really …but

- Invisibility
- Complexity
- Conformity
- Flexibility

make software more problematic to build than other engineered artefacts.

An example here is the Canadian Government Federal pay "Phoenix system."

Complexity/Uncertainty Domain of SDPM

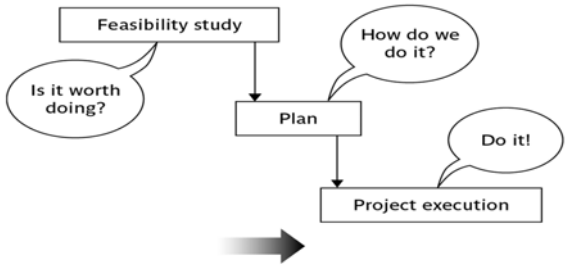
- Complexity and uncertainty are positively correlated with one another. As software development projects become more complex, they become more uncertain. This follows from at least four factors:
  - **Requirements – As project complexity increases, the likelihood of nailing requirements decreases. In 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**

Contract management versus technical project management

Projects can be:

- **In-house**: clients and developers are employed by the same organization
- **Out-sourced:** clients and developers employed by different organizations
- 'Project manager' could be:
  - **a 'contract manager' in the client organization**
  - **a technical project manager in the supplier/services organization**

Activities covered by project management



Feasibility study

Is project technically feasible and worthwhile from a business point of view?

Planning

Only done if project is feasible

Execution

Implement plan, but plan may be changed as we go along

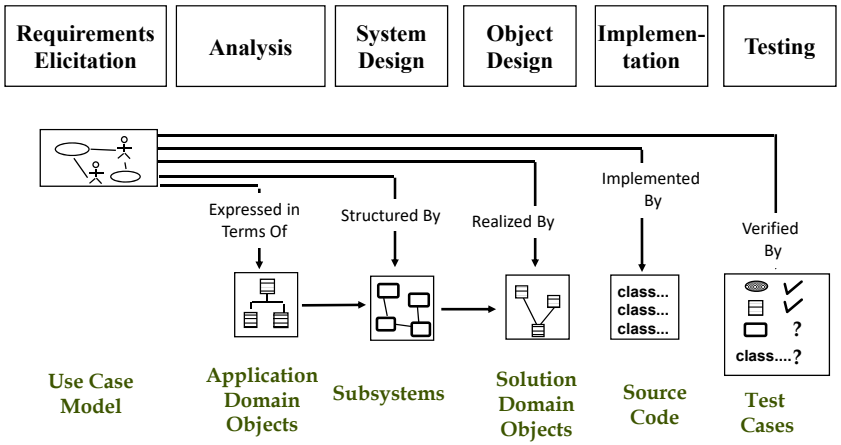
Skills of a good Project Manager

- **Communication Skills:** The single most important characteristic of a first-rated project manager. Written and oral communications are the backbone of all successful projects.
- **Organizational and Planning Skills:** This is important after communication skills. There is need for documentations, requirements information, project reports, personnel records, contracts, etc.
- **Budgeting Skills:** Project mangers establish and manage budgets.
- **Conflict management Skills:** where there is a project, there is a problem. Inter-personal problem solving skills are very important in project management.
- **Negotiation and Influencing Skills:** Effective problem solving requires negotiation and influencing skills. *Negotiation* on project is necessary in almost every area of the project. *Influencing* is convincing the other party that "fish" is better the best fried "chicken." *Power* and *politics* are techniques used to influence people to perform.
- **Power** is the ability to get people to do things they would not do otherwise. It is also the ability to change minds and the course of events.

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 proceedings as planned?**

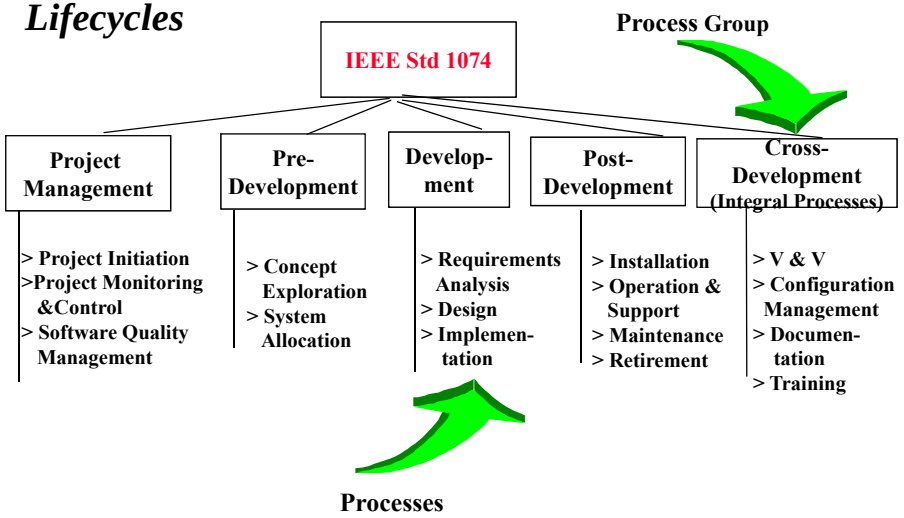
Example of Software Lifecycle Activities



Standard for modeling lifecycles IEEE Std 1074 (Alternative Standard is ISO 12207)

- 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 lifecyle**
    - **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``**

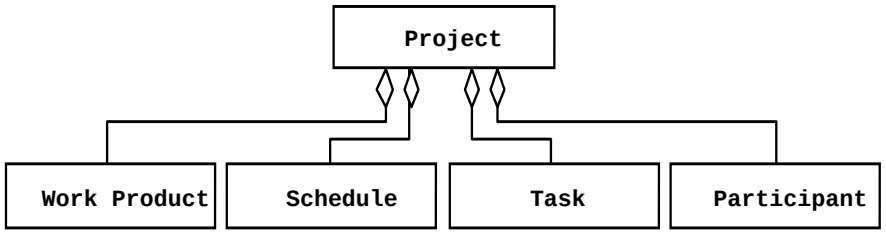
IEEE Std 1074: Standard for Software Lifecycles



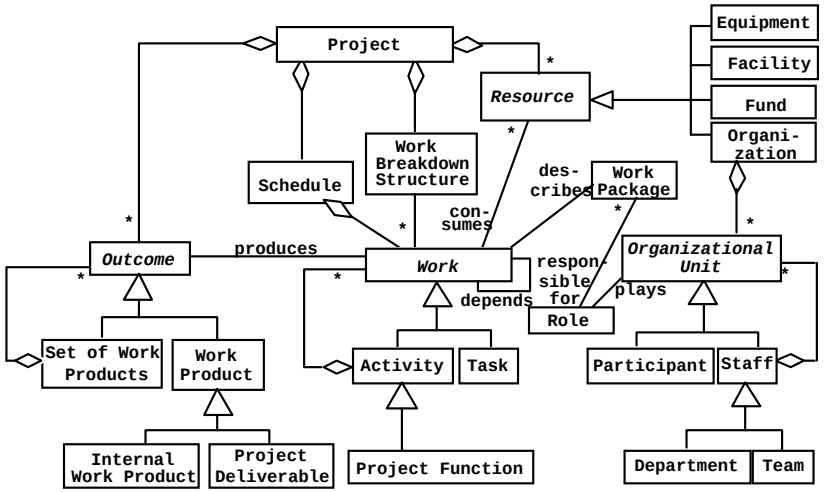
Definitions

- *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.

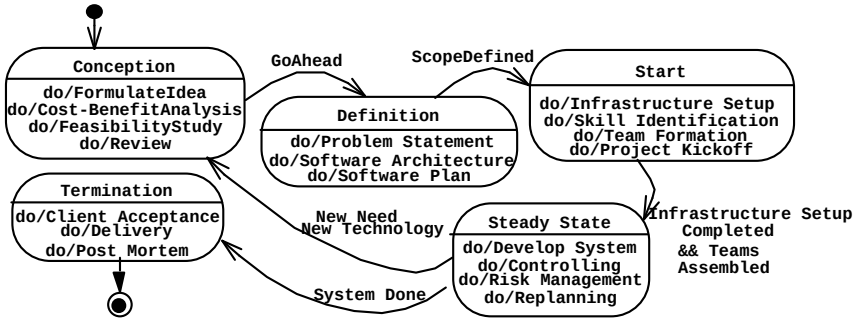
Components of a Project



A More Complex Model



States of a Project



Setting objectives

- Answering the question '*What do we have to do to have a success?*'
- Need for a *project authority*
  - **Sets the project scope**
  - **Allocates/approves costs**
- Could be one person - or a group
  - **Project Board**
  - **Project Management Board**
  - **Steering committee**

Objectives

Informally, the objective of a project can be defined by completing the statement:

The project will be regarded as a success if……..

……

Rather like *post-conditions* for the project

Focus on *what* will be put in place, rather than *how* activities will be carried out

Project Objectives should be SMART

S = Specific, that is, concrete and well-defined

M = Measurable, that is, satisfaction of the objective can be objectively judged

A = Achievable, that is, it is within the power of the individual or group concerned to meet the target

R = Relevant, the objective must be relevant to the true purpose of the project

T = Time constrained: there is defined point in time by which the objective should be achieved

Goals/sub-objectives

These are steps along the way to achieving the objective

Informally, these can be defined by completing the sentence

To reach objective X, the following must be in place

A……

B……

C…… etc

Goals/sub-objectives continued

- Often a goal can be allocated to an individual
- Individual might have the capability of achieving goal on their own, but not the overall objective e.g.

1. *Overall objective* – user satisfaction with software product
2. *Analyst goal* – accurate requirements
3. *Developer goal* – reliable software

Measures of effectiveness

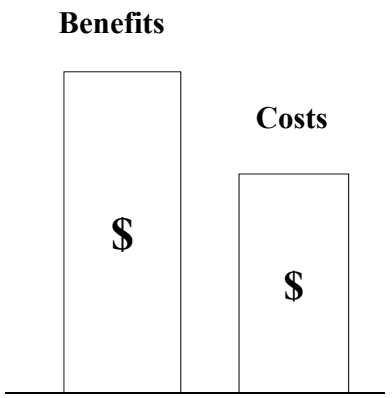
How do we know that the goal or objective has been achieved?

By a practical test, that can be objectively assessed.

e.g. for user satisfaction with software product:

- Repeat business – they buy further products from us
- Number of complaints – if low etc etc

The business case



Benefits of delivered project must outweigh costs

Costs include:

- **Development**
- **Operation**

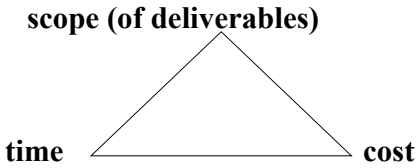
Benefits

Quantifiable

Non-quantifiable

Project success/failure

- Degree to which objectives are met



In general if, for example, project is running out of time, this can be recovered for by reducing scope or increasing costs. Similarly costs and scope can be protected by adjusting other corners of the 'project triangle'.

Other success criteria

These can relate to longer term, less directly tangible assets

- Improved skill and knowledge
- Creation of assets that can be used on future projects e.g. software libraries
- Improved customer relationships that lead to repeat business

What is management?

This involves the following activities:

- **Planning** – deciding what is to be done
- **Organizing** – making arrangements
- **Staffing** – selecting the right people for the job
- **Directing** – giving instructions
- **Monitoring** – checking on progress
- **Controlling** – taking action to remedy hold-ups
- **Innovating** – coming up with solutions when problems emerge
- **Representing** – liaising with clients, users, developers and other stakeholders

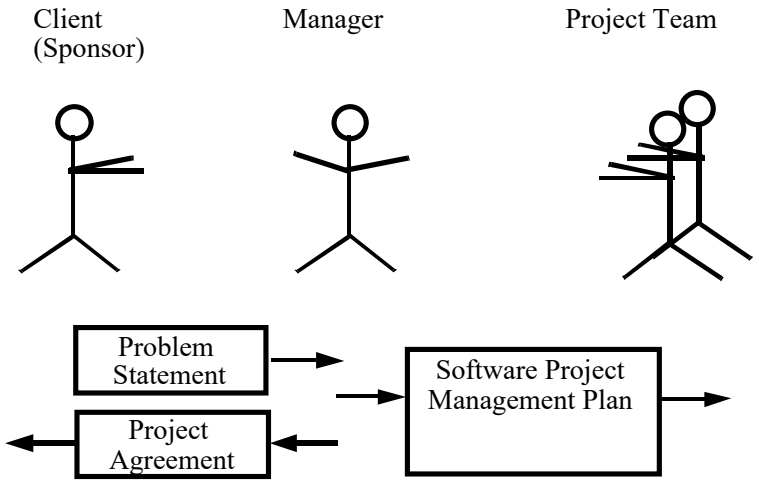
Project Agreement

- 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**

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 .**

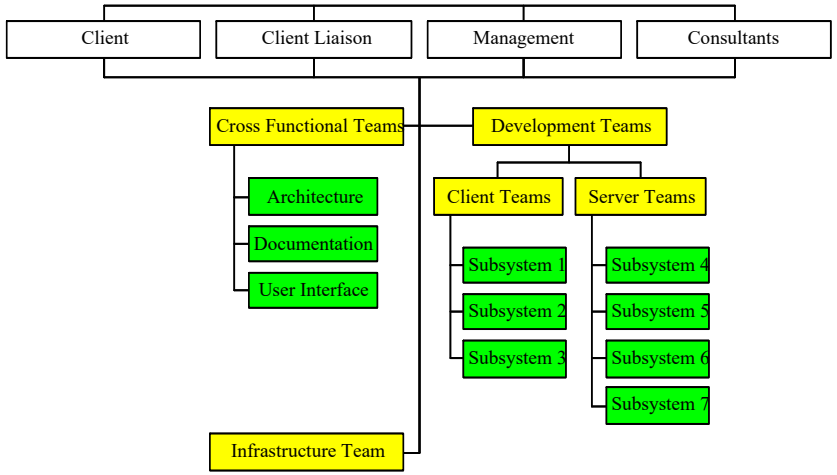
Project Agreement,Problem Statement vs SPMP



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

Organizational Structure Example



Project Management Process Groups

1. **Initiating:** process group occurs at the beginning of each project; acknowledges that a project or next project phase should start; grant approval to commit organization resources to the project. Risk is the highest at this point.
2. **Planning:** planning process group formulate and revise project goals and objective to create project management plan
3. **Executing:** process group involves putting the project plan into action. At this point, the project manager will coordinate and direct project resources to meet the objectives and plan.
4. **Monitoring and Controlling:** process group where project performance measurements are taken and analyzed to determine whether the project is staying true to the project plan. The idea is to identify problems as soon as possible and apply corrective action to control the work of the project.
5. **Closing:** process group brings a formal, orderly end to the activities of a project. This is important because all the project information is gathered and stored for future reference.

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?**

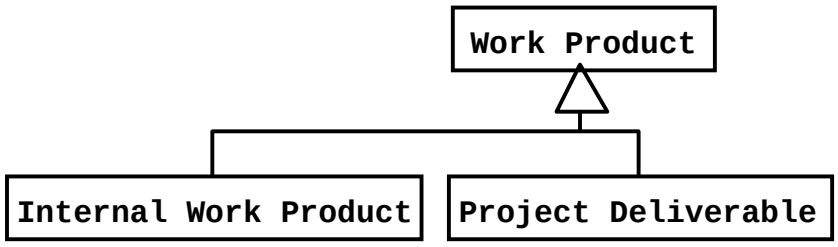
Work package vs Work product

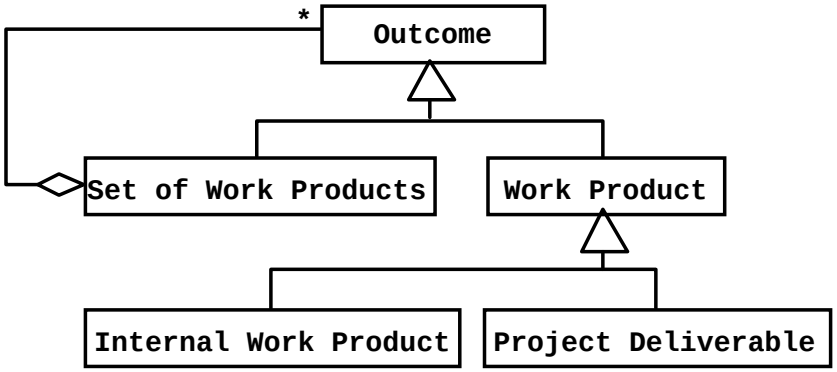
Definitions from the IEEE Standard

- **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?

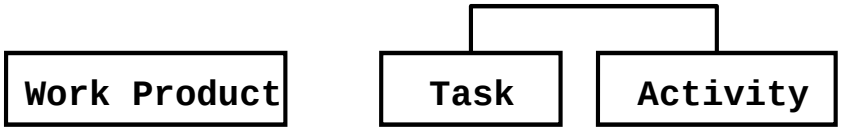
- What is a tangible item?
  - **Requirements Analysis Document**
  - **Software Project Management Plan**
  - **Agenda, Minutes**
  - **How do we model Tangible Items?**





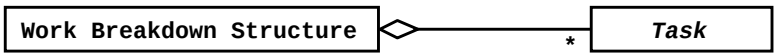
Work products are related to Activities

- How do we model this?

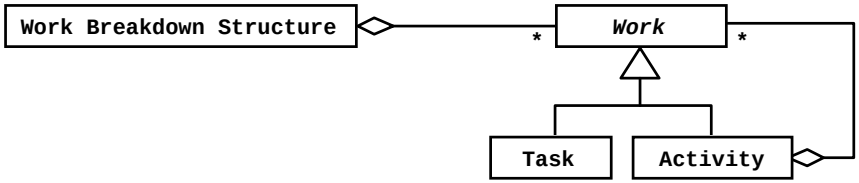


Work Breakdown Structure

- 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)**

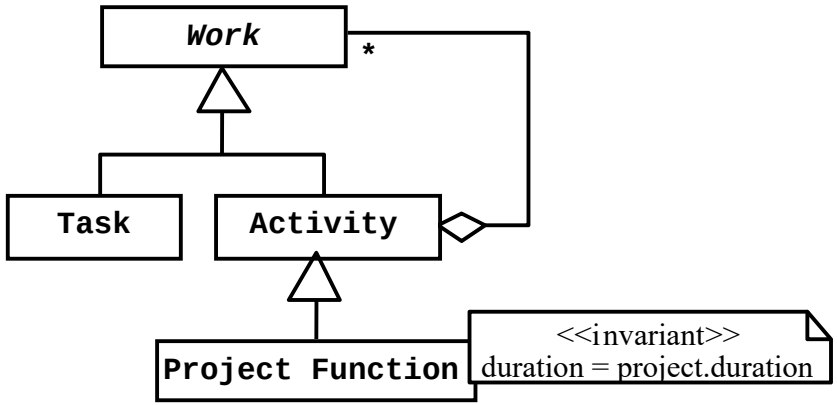
From the WBS to the Dependency Graph

- **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 actitivites, 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



Why are Companies in Business, Anyway?

- The short answer and the most important is to make profit. The so called "Bottom line."
- We may not like the short answer but the truth is that a company that does not make profit cannot survive in the market.
- At times, a company is in the market to minimize loss. An example here is making changes to accounting software as a result of changes in tax law. In this case if the company does not comply with the new tax law may face costly liability. Even in cases like this, the long-term decision is to maximize profit.
- Secondary factors for a company in business include fun, education, social impact, etc..
- Issues such as ethics, concern for customers, environment, etc., may play a part in decision making for a company

Fundamental Concepts of Business Decisions

1. **Proposals** – Making business decision starts with a proposal. A proposal is a single, separate option that is being considered, such as carrying out a particular software development project or not.
2. **Cash-Flow Instance and Cash-Flow Stream** – A meaningful business decision about any proposal must be evaluated using the concepts of cash-flow instance and cash-flow stream.
   a) **A cash-flow instance** is a specific amount of money flowing into or out of the organization at a specific time as a direct result of some proposal.
   b) **The term cash-flow stream** refers to the set of cash-flow instances, over time, which would be caused by carrying out some given proposal.

Fundamental Concepts of Business Decisions

3. **Category of Cash-flow**
   a) **Initial Investment** – This captures all of the one-time, nonrecurring costs associated with starting up a proposal.
   b) **Operation and Maintenance Cost** – These costs occur only after the activity is started and continue through to its retirement.
   c) **Sales Income** – Sales income refers to the direct income generated by the proposal.
   d) **Cost Avoidance** – Cost avoidance is not a direct income but it reduces the expenses needed to produce those products and services. It leaves more of the gross revenue as profit.
   e) **Salvage value** – refers to any remaining value in assets (equipment, facilities, etc.) at the end of the proposal.

"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**