L1: Introduction & Risk Management

Projects, PROJECTS?!

Unlike jobs, which is a repetition of well-defined and well understood tasks, a project is more explorative and is temporary (meaning it has an end).

What is project Management?

- 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 w solutions for problems
- Representing – liaising with clients, users, developers and other stakeholders

Software Projects vs Other Engineering Project

No difference except for:

- Invisibility
- Complexity
- Conformity to "laws of the nature"
- Flexibility

Projects - General

The measurement of a projects success or failure is the degree to which objectives are met.

An example would be a project running out of time, which could be recovered by:

A. Reducing the scope of the project. which would in turn reduce the lead time (total time for the project from start to finish)

B. Increasing the costs/spent effort by adding more resources to the resource pool

On average, a software project costs 30% more than planned. 1/6 of IT projects are 'black swans' meaning it costs 200% more than planned and takes 70% more time than planned.

Software Project Management - Main Areas

- Activity planning
- Effort estimation
- Risk management
- Resource allocation
- Monitor & control execution

Risk Management

> *A risk is a potential problem; a problem is a risk that has materialized*

[Fairley 1994]

What can go wrong?

- Staff turn-around: quit, moved to other project, illness
- Tools do not work as anticipated
- Requirements change more than expected
- Effort estimates were wrong
- Expected input/deliveries are delayed
- Quality is not good enough in the end of the project

What is risk?

"An uncertain event or condition that, if it occurs, has a positive or negative effect on a project's objective" It consists of both a **cause** and an **effect** For example:

- Use of new compiler (**cause**)
    - Integration problem && delayed training phase (**effect**)
- Low top management priority in Project A (**cause**)
    - Resources (staff) moved from project A to another project (**effect**)

Risky commitments - Overscoping

The main risk of overscoping is that the project target (scope) will not be met in *time* and on budget (*cost/effort*).

- The traditional dev model (waterfall) - sequential & doc-driven
    - "*overpromise* software capabilities in cortractually binding requirements specification before they *understand* their risk implications."
- Agile dev model (XP/Scrum) - iterative & code driven
    - "... neat ideas... I'll code 'em up, and if they *don't fit* other peoples ideas, we'll just evolve things until they work."

Risk Management

- Risk Assessment
  - Identification (What might go wrong?)
  - Analysis & prioritization (What are the most serious risks?)
- Control of risk
  - Planning & Resolution
  - Monitoring

Approaches to risk identification:

Includes but isn't limited to:

- Use of checklists, check which risks may come up based on previous experience.
- Brainstorming - getting knowledgeable stakeholders together to pool concerns.
- Casual mapping - Identifying possible chains of cause and effect using a map.

Causal mapping (Applied example)

```
                                (DELAY IN PROJECT!)

                               /                   \
                              /                     \
                             /                 (People need education)

                            /                             \
                           /                               \
                          /                                 \
(Unrealistic schedule estimates)                      (Lack of skill)
```

Risk assessment

Risk exposure = Potential Damage * Probability

- Not neccesary nor possible to give exact estimates: Qualitative descriptors , e.g High, Significant, Moderate and Low.
- Pripritizethe worst risks (high probability and large damage)

METHOD: Cost-Probability Matrix

```
(cost)        _____________________________
             |       |xxxxxx|xxxxxx|xxxxxxx|
High         |       |xxxxxx|xxxxxx|xxxxxxx|
             |_______|xxxxxx|xxxxxx|xxxxxxx|
             |       |      |      |xxxxxxx|
Significant  |       |      |      |xxxxxxx|
             |_______|______|______|xxxxxxx|
             |       |      |      |       |
Moderate     |       |      |      |       |
             |_______|______|______|_______|
Low             |       |      |      |       |
                |       |      |      |       |
             |_______|______|______|_______|
                Low    Med    Med+   High    (probability)
```

The worst risks are indicated by X'es since they both have a high cost and probability.

METHOD: Decision Tree

In this scenario we ponder on wether or not to extend or replace a system. The outcome of this depends on if the market expands or not. If we extend the system and the market expands we lose -100.000$, if it doesn't expand we will gain 75.000$. If we replace the system and the the market expands we gain 250.000$, if it doesn't expand we lose 50.000$

```
                          ,_______________      Net Product Value (NPV)

           ,____________/  (20%) Expansion     -100.000 $

          /    Extend    \ (80%) No Expansion   75.000 $

         /                `________________
        /
       /
D
       \
        \
         \                ,_______________
          `_____________´  (20%) Expansion      250.000 $

               Replace   \ (80%) No Expansion  -50.000 $

                          `________________
```

If we extend the risk exposure is: RE = -100.000 ***0.2 + 75.000*** 0.8 = 40.000 And if replace the risk exposure is: RE = 250.000 ***0.2 - 50.000*** 0.8 = 10.000 Therefore in this example we should obviously **EXTEND THE SYSTEM!!!!!!**

Risk planning: There are five alternatives:

AARMT

- **A** cceptance
- **A** voidance - Find a risk-free solution.
- **R** eduction - Reduce probability
- **M** itigation - Reduce damage, e.g. taking backups
- **T** ransfer - e.g. outsource

METHOD: Risk reduction leverage

Risk reduction leverage is a method of comparing different options by comparing risk exposures.

RRL = (RE_before - RE_after)/(cost of risk reduction)

RE_before is risk expose before risk reduction, e.g. 1% chance of a fire causing $200k dmg.

RE_after is risk exposure after risk reduction, e.g. $500 alarm which reduced probability of fire dmg by 0.5%

RRL = (1% of $200k - 0.5% of $200k)/$500 = 2 RRL > 1.00 *therefore worth doing*

The risk of delays in completing activities, examples.

Risk
Personell shortfalls
Unrealistic time and cost estimates
Developing the wrong software functions
Developing the wrong user interface
Gold plating
Late changes to requirements
Shortfalls in externally supplied components
Shortfalls in externally performed tasks
Real time performance problems
Development technically too difficult

METHOD: Using PERT

PERT = Program Evaluation and Review Technique

PERT - A statistical tool for analysing completion time

Method

Three estimates are produced for each activity (task)

- (m) Most likely time
- (a) Optimistic time
- (b) Pessimistic
- (t_e) 'expected time' t_e = (a +4m +b)/6
- (S) 'activity standard deviation' S = (b-a)/6

Calculations

Suppose the dependecies are as such:

```
 ______      ______      ______
|TASK A| -> |TASK B| -> |TASK C|
 ¨¨¨¨¨¨      ¨¨¨¨¨¨      ¨¨¨¨¨¨
```

Task	a	m	b	t_e	s
A	10	12	16	13	1
B	8	10	14	10	1
C	20	24	38	26	3
A+B+C				49	3

The calculations are completed. Just use the previously mentioned formulae if you want to review them.

Assessing the likelihood of meeting a target

- Imagine now that the target for completing A+B+C was 52 days.
- Calculate the Z-value as (T - t_e)/s = (52 - 48.65) / 3.32 = 1.01
- Match the Z-value with a probability by using a pre-existing table.

Which results in a 15% chance of **NOT** meeting the target 52 days. And that's PERT for you, kids!

Problems with estimates of task duration

- Estimators usually add a safety zone to cover difficulties
- Developers work to the estimate, meaning the time is lost
- No advantage is taken of opportunities where tasks can finish early - and provide a buffer for later activities

  Parkinson's law:

  Work expands so as to fill the time available for its completion

  => All buffers are usually consumed by end of the project.

Basic Ideas of Critical Chain

To reduce time wasted, a critical chain is constructed as such:

```
 _____ __
|_____'__| ____ __
          |____'__| ___ __
                   |___'__|
```

Move all buffers to the end and halve them.

```
 _____
|_____|  ____
       |____| ___ ___
             |___'___|
```

Thus, the buffers have been reduced in half so that people don't waste time!!!

Critical Chain Approach

1. Ask the estimators for two estimates
    - Most likely duration: 50% chance of meeting this
    - Comfort zone: additional time needed to have 95% chance
2. Schedule all activities using most likely values and starting all activities on latest start date
3. “Critical chain” the same as “critical path” but resources also considered
4. Put a project buffer at the end of the critical chain with duration 50% of sum of comfort zones of the activities on the critical chain
5. During project execution monitor how much of the buffer that has been used
6. Supported in tools, e.g. through add-on to MS Project

Executing and monitoring Critical Chain plans

- Principle: focus your efforts - ”multitasking is evil”
    - No chain of tasks is started earlier than scheduled, but once it has started is finished as soon as possible
    - This means the activity following the current one starts as soon as the current one is completed, even if this is early – the relay race principle
- Fever charts are used to monitor progress and catch tasks at risk

Summary of Risk Management

- Definition of ‘risk’ and ‘risk management’
- Risk management
    - Risk identification – what are the risks to a project?
    - Risk analysis – which ones are really serious?
    - Risk planning – what shall we do?
    - Risk monitoring – has the planning worked?
- Methods: causal mapping, probability impact matrix, decision trees
- Managing risk of delay with PERT and critical chain

Activity planning

Time is natures way of stopping everything happening at once.

The objectives of activity planning

- Feasability assessment
    - Is the project possible within the required timescales and resources?
- Resource allocation
    - What are the most effective ways of allocating resources? When should the resources be available?
- Detailed costing
    - How much will the project cost? After producing an activity plan, we can obtain more detailed estimates of costs.
- Motivation
    - Providing targets and monitoring progress can motivate staff.
- Coordination
    - When do the staff need to be transfered between projects to increase efficiency?

Activity networks

```
           ____
          |____|
    ____ /      \ ____
-->|____|        |____|-->
         \ ____ /
          |____|
```

Activity networks can be used to assess the feasability of the projects completion date, identify when resources are needed, calculate when costs incur, and to coordinate the staff between tasks.

Defining activities

An activity includes:

- Start and end-time
- Resource requirements

- Duration
- Dependencies

Identifying activities

There are three approaches to identifying the activities or tasks:

- Activity based planning
    - Creating a list of all the activities that the project needs. Can be done by creating a **Work Breakdown Structure** (WBS, filled with verbs). This involves identifying the main tasks that are needed for the project and then breaking them down into lower-level tasks. Too great depth will result in a large number of small tasks that will be diffcult to manage. Too shallow project provides insufficient detail.
- Product-based approach
    - Consists of producing a **Product Breakdown Structure** (PBS, filled with nouns) and a **Product Flow Diagram** (PFD). The PFD idicates for each product which other products are required as inputs. With the help of a PFD you can easily create an ordered list of activities by identifying the order of products and which acitivies are needed for them.
- Hybrid approach
    - The hybrid approach is a mix of both. Instead of creating a WBS that is based on the projects activities, you create a WBS that is based on the projects products.

Network planning models

- Program Evaluation Review Technique (PERT)
    - Activity on arrow
- Critical Path Method (CPM)
    - Activity on arrow
- Precedence network
    - Activity on node
    - most common

Rules for precedence networks:

- Should have only one start node.
- Should have only one end node.
- Every node has a duration.
- Links normally don't have a duration.
- Precedence are the immediate preceding activities.

- Time moves from left to right.
- A network may not contain loops.
- No dangling activities.

Lagged activities

```
  ____         ____         ____
 |____|----->|____|----->|____|
           1/  ____  \2
            \|____|/
```

Lagged acitivites are dependent on other activities but are initiated with a lag. In the above example, the second-row activity is started 1 day after the dependent activity and completed 2 days after the dependent activity has completed.

Start and finish time of activities

- Earliest start (ES)
- Earliest finish (EF) = ES + duration
- Latest finish (LF) = latest task that can be completed without affecting project end.
- Latest start = LF - duration

Forward and backward pass

- Forward pass
  - A forward pass calculates the earliest dates on which each activity may be started and completed.
  - It does this by going through the dependencies from start to finish and sorting the activities accordingly.
- Backward pass
  - A backward pass calculates the latest date at which each activity may be started and completed.
  - It does this by going backwards and looking at the duration of each activity.

Critical path and float

The float of an activity is defined as the difference between its earliest start date and latest start date (or earliest finish - latest finish). It is a measure of how much the start or completion of an activity may be delayed without affecting the end of the project.

- Free float

        - The time by which an activity may be delayed without affecting any subsequent activities.
    - Interfering float
        - The difference between total float (Vad är total float, berätta för mig / Andrée) and free float. Indicates how much the activity can be delayed without delaying the project end date.

The critical path shows the path of activities in a project that define the duration of the whole project. Any delay in these activities will delay the whole project, which means these activities have no float.

If a project duration needs to be reduced it is the critical path that needs to be shortened.

Effort Estimation

Problems with Effort Estimation

Subjective nature of estimating

- Difficult to produce evidence in support of decision
  - these bring uncertainties, especially in the early days when there is a 'learning curve'
- Projects differ
  - Experience on one project may not be applicable to another
- Political pressure
  - Managers may wish to reduce estimated costs in order to win support for acceptance of a project proposal
- Covering your own back
  - Avoid later overtime

Effort or Cost?

```
Effort == work required e.g. person/man-hours
```

Cost includes:

- Person-hours (salaries, benefits)
- Moving and living costs for new staff members
- Training
- Travel costs
- Legal & Licensing fees (software, patents, copyright)
- Contractors and subcontractors
- Equipment
- Currency exchange rates
- Marketing & Advertising
- Inflation rate

Over- and under-estimating

Parkinson's Law

> *Work expands to fill the time available*

Brooks's

> *Putting more people on a late job makes it later*

Weinberg's Zeroth Law of reliability

> *a software project that does not have to meet a reliability requirement can meet any other requirement*

SVENSKA INCOMING

Vad betyder då den här lagen? Jo det ska ni få veta! (Brace yourselves) Lagen innebär att ifall projektet inte har en tydligt specificerad krav på tillförlitlighet när det ska skeppas så kan detta arbiträrt väljas. Vilket resulterar i sämre kvalitet som sedan visas i testning eller senare faser.

- Over-estimations => project likely to take longer
- Under-estimations => lower quality in order to meet target

When & Why

- Strategic planning
- Feasability study
- Requirements spec
- Evaluation of suppliers' proposals
- Project planning

What don't we know?

- Customers' needs & expectations - requirements
- Technical complexity & design components
- Reuse - internal and external components
- People
  - Skills, competence and experience
  - Co-operation & communication
- ... (srsly wtf, why the dots here grrl?)

Bottom-up versus top-down estimation

Bottom-up

- Identify all tasks that have to be done (top-down)
- Estimate tasks of 1-2 mw, accumulate effort bottom-up
- Time consuming
- Useful when no past project data for similar projects exist

Top down

- Produce overall estimate based on project cost drivers
- Based on past project data
- Divide overall estimate between jobs done

```
effort = (system size) * (productivity rate)
```

Estimation methods

- Analogy: case-based, comparative (from previous projects etc)
    - Look at previous similar tasks, base estimation from diff/error
- Parametric or algorithmic models, e.g. function points COCOMO
- Expert opinion - just guessing?
- Parkinson and 'price to win' - opportunistic estimates!

Basis for successful estimation

- Information about past projects
    - Need to collect performance details about past project
- Need to be able to measure the amount of work involved
    - Traditional size measurement for software is 'lines of code' - ain't good

Stages: Identify

1. Significant features of the current project
2. Previous project(s) with similar features
3. Base effort on previous similar project. Consider
    1. Differences between current and previous projects
    2. Possible reasons for error (risk)
    3. Measures to reduce uncertainty

Algorithmic estimation

Estimates based on historical data in the form of measurements from earlier projects, typically `effort = c * size^k`

Example:

```
size = lines of code
1/c  = productivity //e.g. lines of code per day
```

- Objective, but good estimate requires
  - Good experience base
  - Good size estimate

Parametric Models

Used in top-down approaches. Productivity factors are used as parameters. A parametric model normally has formulae in the form

```
effort = (system size) * (productivity rate)
```

where system size may for example be measured in KLOC (kilo-lines of code) or function points and productivity rate in days per KLOC.

Some parametric models are focused on system or task size, while others like COCOMO focus on productivity factors.

Function Points - Albrecht/IFPUG FP

Models system size

```
UFP = SUM(i=1..15, n items of weight i * weight)
FP   = UFP * "technical complexity factor"
```

Albrecht function point analysis

- Top down method developed by Allan Albrecht, IBM.
- Information systems comprised of five ***external user types***:
  - *External input types* – input transactions which update internal computer files
  - *External output types* – transactions where data is output to the user
  - *External inquiry types* – transactions initiated by the user, which provide information but do not update internal files
  - *Logical internal file types* – are the standing files (or datastore) used by the system.
  - *External interface file types* – allow for output and input that may pass to and from other computer applications

- A table consists of "low", "medium" and "high" complexity numbers for the above listed points. The amount of items in each category is then multiplied by its counterpart in the table. The weighted sum of these are the Albrecht FPs.

COCOMO (COnstructive COst MOdel)

Developed by Barry W. Boehm. Focuses on productivity factors. The first version (COCOMO81) used the formula

```
effort = c * size^k
```

where effort is measured in person-months and size in KLOC. c and k are constants that depend on the system being developed. The exponent k is selected to cause larger projects to require disproportionately more estimated effort than smaller ones in the es, as larger projects are often found to be less productive in practice.

In COCOMO II there are models for estimates at three different stages in a project; ***Application composition (inception)***, ***Early design (elaboration)*** and ***Post architecture (construction)***. The core model is now

```
pm = A * (size)^(sf) * (em_1) * (em_2) * ... * (em_n)
```

where A is a magical constant and sf, the scale factor, depends on the following ***exponent driver ratings***:

- Precedentedness (PREC)
- Development flexibility (FLEX)
- Architecture/risk resolution (RESL)
- Team cohesion (TEAM)
- Process maturity (PMAT)

These ratings are estimated on a six level scale ranging from ***very low*** to ***extra high***. From this guesstimation a table provides a numerical magical constant.

em_1..em_n are ***effort multipliers***. A wide range of standard effort multipliers are defined in COCOMO II for every development stage along with corresponding magical constant.

Expert judgement

Ask an important looking person to pull a suitable guesstimation out of his pocket. Research show that expert judgement in practice often is based on ***analogy***.[citation needed]

How to? First, find the right people (people with experience from similar products and processes). Then, ask the right questions (present context and goals, be specific about resource restrictions). Finally, ensure the experts have time to think about it.

Software metrics

You cannot control what you cannot measure

[some random dude named Tom DeMarco 1986]

- SW project management
    - Estimate cost/effort based on historical data
    - Monitor project progress & risks
- Software quality – product quality status
- SW process improvement – process efficency & quality

All of these require decision support!

Examples

- Products
    - Lines of code: sw size, impl efficiency
    - Size of requirements specification
    - Complexity of design
    - Test coverage of code
- Processes
    - Actual implementation time
    - Average time for issue resolution
    - Number of issues submitted by testers
    - Effort spent in design phase
- Resources
    - Price of software components
    - Experience of staff

Objective or subjective measurments

- Objective
    - Lines of code
    - Actual implementation time – source: reporting system
    - Module complexity – cyclomatic complexity
- Subjective

    - Experts or questionnaires with Likert scales (ordinal)
    - Complexity of system
    - Usability of system

Scales

- Nominal scale – The nominal type, sometimes also called the qualitative type, differentiates between items or subjects based only on their names or categories and other qualitative classifications they belong to. Ex. Gender or language.
    - type of fault = {data, function, interface, …}
- Ordinal scale – The ordinal type allows for rank order (1st, 2nd, 3rd, etc.) by which data can be sorted, but still does not allow for relative degree of difference between them.
    - criticality of fault = critical > important > medium > low
- Interval scale – The interval type allows for the degree of difference between items, but not the ratio between them. Ex. direction measured in degrees from true or magnetic north.
    - end time = date
- Ratio scale – The ratio type takes its name from the fact that measurement is the estimation of the ratio between a magnitude of a continuous quantity and a unit magnitude of the same kind. A ratio scale possesses a meaningful (unique and non-arbitrary) zero value. Most measurement in the physical sciences and engineering is done on ratio scales.
    - duration = days

Scales: allowed usage

- Nominal scale: labelling, classifying entries
- Ordinal scale: order entries
- Interval scale: relative distance between entries
- Ratio scale: relative size

Metrics challenges

Challenges

- Data collected in different projects should be comparable
- Data collected should be reliable and of high quality
- People do not want to collect data that is not used

Solutions

- Collect only data that you really need (but all data that you need)

- Understand and record the context in which data is collected
    - Product
    - Team
    - Process
    - etc.
- Quality assurance for data collection

Goal question metrics (GQM)

Top-down approach

- Conceptual level – Goals What are we trying to achieve? Example: shorten lead time of issue management
- Operational level - Questions about areas of uncertainty related to the goals. You need process knowledge to derive these. Example: average time to resolve issues?
- Quantitative level – Metrics Measurements that answer the questions Ex: time(open → closed) in issue management system

Examples

Goals

Purpose - improve
Issues (quality focus) - the efficiency
Object(s) - issues management
Viewpoint - from the project manager's viewpoint

Align with

- Organisational policy & strategy
- Relevant processes / products
- Organisational structure for viewpoint

Questions

1. How characterize the object, i.e. issue management? *Q1: What is current turn-around time for issues of varying prio?*
2. How characterize attributes relevant to issues, i.e. issues management efficiency? *Q2: What is the deviation in turn-around time compared to committed response time?*
3. How evaluate characteristics of object relevant to issues & viewpoint? *Q3: Is the current performance satisfactory from project manager's viewpoint?*
   Q4: Is the performance visibly improving?

Measurements

Q/m	Info
Q1	**What is current turn-around time for issues of varying prio?**
M1	*Average cycle time (open → close), for all prio types*
M2	*Standard deviation*
M3	*% cases outside the upper limit, per prio type*
Q2	**What is the deviation in turn-around time compared to committed response time?**
M4	*(Current average cycle time (M1) – Committed response time)/ M1*
M5	*Subjective evaluation by project manager*
Q3	**Is the current performance satisfactory from project manager's viewpoint?**
M5	*Subjective evaluation by project manager*

Summary

- SW Metrics provide decision support
  - Project progress, product quality, process, cost etc.
- Objective vs subjective measurements
- Different scales
- Goal Question Metrics (GQM): top-down approach

Software Process Improvement

- Why focus on processes?
    - Processes affect **product quality**
    - Processes affect **people**, **schedule** and **technology**
    - Software is more design intense => people dependent

Software processes are different from manufacturing processes in that:

- Manufacturing is ***repeatable*** -- the same item is produced many times, while software is ***unique*** -- it is written only when nothing similar exists.
- Manufacturing is ***well-defined*** -- the specification of the output is clearly defined before the process starts, while software is ***incomplete & evolving*** -- requirements are sketchy from the start and also likely to change during the development cycle.
- Manufacturing is ***known*** -- the process between input and output is clearly known and can be done repeatedly, while software is ***unknown*** -- it is based on new technologies, new interfacing requirements etc.
- Manufacturing is based on ***machines & tools*** -- the process' efficiency is mostly dependent on the quality of the machines and less on the operators, while software development is based on ***people*** -- knowledge, experience and skill of people can make huge differences in productivity.

Process Management for Software

- **CMMI:** Capability Maturity Model Integration, software certification similar to ISO
- **PSP:** Personal Software Process
- **TSP:** Team Software Process

Personal Software Process

> *The PSP aims to provide software engineers with disciplined methods for improving personal software development processes.*

- Goes from PSP0 => PSP2.1
- PSP0, PSP0.1: Introduces process discipline and measurement
    - 3 phases: planning, development and post mortem
    - Baseline established: time spent programming, faults injected/removed, size of program
    - In PSP0.1 a coding standard is introduced, size measurements and PIP

    - PIP - engineer establishes a personal improvement plans and records ideas on self-improvement
- PSP1, PSP1.1: Introduces estimating and planning
    - Based on the baseline established in PSP0 the engineer prepares a test report. Data used is accumulated from the baseline data and used to estimate total time.
    - Each new project will record actual time spent, this information is used for task schedule planning. (PSP1.1)
- PSP2, PSP2.1: Introduces quality management and design
    - PSP2 adds two new phases: design review and code review. Defect prevention and removal are the focus at the PSP2. Engineers learn to evaluate and improve their process by measuring how long tasks take and the number of defects they inject and remove in each phase of development. Engineers construct and use checklists for design and code reviews. PSP2.1 introduces design specification and analysis techniques

Team Software Process

Before engineers can participate in the TSP, it is required that they have already learned about the PSP, so that the TSP can work effectively. Training is also required for other team members, the team lead, and management.

The TSP software development cycle begins with a planning process called the launch, led by a coach who has been specially trained, and is either certified or provisional. The launch is designed to begin the team building process, and during this time teams and managers establish goals, define team roles, assess risks, estimate effort, allocate tasks, and produce a team plan. During an execution phase, developers track planned and actual effort, schedule, and defects, meeting regularly (usually weekly) to report status and revise plans. A development cycle ends with a Post Mortem to assess performance, revise planning parameters, and capture lessons learned for process improvement.

The coach role focuses on supporting the team and the individuals on the team as the process expert while being independent of direct project management responsibility. The team leader role is different from the coach role in that, team leaders are responsible to management for products and project outcomes while the coach is responsible for developing individual and team performance.

Software process improvement

1. Evaluation of current practices
2. Planning for improvements
3. Imlementing improvements
4. Evaluation of effects

The Deming Cycle - PDSA

```
          , - ~ ~ ~ - ,
      , '       |       ' ,
    ,           |    Act    ,
   ,    Study ->|     |      ,
  ,             |     v       ,
  ,-------------|-------------,
  ,       ^     |             ,
   ,      |     |<-  Plan    ,
    ,    Do     |           ,
      ,         |        , '
        ' _ , _ _ _ ,  '
```

These steps are also known as the "*General Steps for SPI*" (Software process improvement)

- Study - Study/Check/Assess the outcome; measure and report
- Act - Decide on needed changes -> repeat
- Plan - Plan goal & process design/revision to improve results
- Do - Implement plan & measure performance

SPI approaches

Prescriptive

Top-down approaches (general -> specific)

Examples: CMMI, SPICE

Inductive

Bottom-up approaches (specific parts -> general)

Examples: QIP, iFLAP, Lean Six Sigma

- Process modelling and simulation
- Information flow analysis
- Retrospective reflection aka Lessons learnt, project post-mortem

Lean Six Sigma

- **Focus:** elimination of waste in the process flow (LEAN)
- **Combination** of **LEAN** and **Six Sigma**
  - *Lean:* Toyota productions, eliminate waste, quality
  - *Six Sigma:* Motorola (1986) 99.99966% of products statistically expected to be free of defect (6σ)
- A process for process improvement

- LSS 'belts' - training and certification

LSS Process for Process Improvement (yo dawg I heard you liek process improvement in your process improvement)

```
Entry     * Prepare charter, sponsor, team and leader
  |
  V
Define    * The problem
  |       * The relevant process
  V       * Customer critical aspects
Measure   * Current performance
  |
  V
Analyze   * Data and process
  |       * Identify root causes
  V
Improve   * Process w solutions
  |
  V
Control   * Ensure targets met over time
  |
  V
Exit      * Share results, benefits and lessons learnt
```

Define: input, process factors, output **Measuring**: Lead time, customer satisfaction, cost per customer, competence level etc **Analyze**: root cause analysis, "5 Why?", identify improvement **Improve**: implement **Control**: remeasure, continously assess

Retrospective Analysis

- Consider the past in order to identify problems and improvements - individual, but mostly in groups
- Often applied after project completion
- Important SPI method for self-governing agile teams
  - Sprint (iteration) retrospectives
- **Benefits**: Team learning & improvements, widen perspectives & insight into bigger picture
- **Challanges**: taking the time together, remembering (correctly and uniformly), risk of incorrect conclusion for pure experience based retros

Evidence-Based Timeline Retrospectives A PM extracts *evidence* from the project, i.e. notes, tasks, estimations etc, to form a Timeline. The team, or parts of the team, are then invited to review the timeline. This gives a chance for reflection and stimulates discussion around the project. It also gives the team a chance to jointly identify issues and solve them in a positive manner. The aim of the meeting is to further analyze how different events and actions influence each other with the aim of identifying practices that work or that need improvement.

Roles: Moderator (leading discussion and creating positive atmosphere), co-moderator (basically secretary), team member (discusses and evaluates)

CMM & CMMI

Capability Maturity Model for Software (SW-CMM)

- Mission to promote software technology transfer, particularly to defense contractors
- Maturity model proposed in mid-80s and later refined to **CMMI** in early 90s
- Work has been very influential in process improvement

CMMI was introduced to bring together models produced for different environments into a single integrated framework. These models place organizations at one of five levels of process maturity which indicate the sophistication and quality of their production practices. These levels are defined as follows.

Structure

- Maturity Level: indicate capability and contains Key Process Areas
- Key Process Areas: goals and common features
- Common Features: addresses implementation and contains Key Practices
- Key Practices: describes infrastructure and activities

Maturity Levels

- **Level 1** *Initial*: The procedures followed tend to be haphazard. Some projects may be successful, but this tends to be because of the skills of particular individuals, including project managers. There is no level 0 and so any organization would be at this level by default.

  - Frequently difficulties in making commitments
  - Crises common
  - Success depends entirely on having exceptional managers and developers
  - Level 1 companies, however, can deliver products

  Key Process Areas

  - None, initial level

- **Level 2** *Managed (Repeatable, in slides)*: Organizations at this level will have basic project management procedures in place. The way, however, individual tasks are carried out will depend largely on the person doing it.

  - Policies for managing software projecs are implemented
  - Realistic commitments
  - Capability: disciplined, earlier successes can be repeated

  Key Process Areas

  - Configuration management

    - Software QA
    - Subcontract management
    - Project tracking and oversight
    - Project planning
    - Requirements management

- **Level 3** *Defined*: The organization has defined the way that each task in the software development life cycle should be done.

    - A typical process for developing and maintaining software in the organisation is defined
    - A Software Engineering Process Group (SEPG) is defined
    - Capabilitiy: standard and consistent - both software engineering and management are stable and repeatable

  Key Process Areas

    - Organization process def
    - Peer reviews
    - Training program

- **Level 4** *Quantitatively managed*: The products and processes involved in software development are subject to measurement control.

    - The organization sets quantitative quality goals for both products and processes
    - Software products are of high predictable quality
    - Organization-wide metrics database
    - Meaningful variations can be distinguished from noise
    - Capability: predictable

  Key Process Areas

    - Software quality management
    - Quantitative process management

- **Level 5** *Optimizing*: Improvement in procedures can be designed and implemented using the data gathered from the measurement process.

    - The whole organization is focused on continous process improvement
    - The organisation has the means to identify process weaknesses and take actions
    - Cost benefit analysis possible

  Key Process Areas

    - Process change management
    - Technology change management
    - Defect prevention

The evaluation is performed by a team of assessors coming into the organization and interviewing key staff about their practices, using a standard questionnaire to capture the information. A key objective is not just to assess, but to recommend specific actions to bring the organization up to a higher level.

- Contains 25 process areas, e.g. requirements management, quality assurance etc
- Each process area contains goals and practices
- Two types of models
    - Staged: grades the overall development process
    - Continous: grades each process area

SPICE

Software Process Improvement Capability dEtermination model

- Developed from CMM
- Parts of SPICE defined as ISO standard
- Consists of:
    - Process dimension: assessment per process area
    - Capability dimension: how processes are impl and managed
- Particularly appropriate for small organisations (ability to focus on process areas)

Process Modelling

- Map & Understand
- Facilitate group communication
- Process guidance and tool support for process flows

Simulating processes

- What-if analysis
- Measuring of flow for different alternatives
- Requires tool support

Information Flows

- Focuses on flow and transformation of information e.g. requirement changes to testers, tech dep vs comm flow (?)
- Used to identify bottlenecks, missing connections, information brokers (key nodes)
- Social network analysis often applied

Change requires...

- Commitment from management
- Support & training
- Staff involvement
- Measurable goals

SMART Goals

- Specific - who, what, when, where, why, how?
- Measurable - How will you know when you're done?
- Attainable - Is this realistic?
- Relevant - How does this relate to current situation & visions for future?
- Time-bound - what's the deadline?

Agile project management

Agile is a name for many different methods, each of these implementation is unique. The included methods are Scrum, XP, and Kanban.

The Agile manifesto

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

Agile projects

Traditional development:

```
|Requirements->|Design->|Implementation->|Testing->| Done!
```

Agile development:

```
|R->|D->|I->|T->|
                |R->|D->|I->|T->|
                                |R->|D->|I->|T->|
                                                 Done!
```

Agile focuses on...

- Maximizing return on investment (ROI), delivering business value.
- Quickly delivering working code.
- *Expecting* and managing changes.

Agile SPM involves gradual detailing, prioritized work order that are just-in-time, and independent teams.

Exploration

- Define high-level requirements
- Prioritize from business perspective
- Explore / prototype

Planning

- Prioritize user stories
- Backlog (scrum)
- Story cards, story board, Kanban board
- Re-estimate, for each sprint
- Select set of stories for iteration according to capacity

Kanban board

Simplest version - columns for to-do, in progress, done

Common version - Backlog, Ready, Coding, Testing, Approval, Done columns

Used to, visualize workflow, limit work-in-progress, pull work from column to column, monitor, adapt, improve etc.

Time boxing

- Time-box fixed deadline by which something has to be delivered
- Typically two to six week iterations
- Develop the most prioritized stories first

Development

- Iterations a.k.a. sprints
  - Scrum: 30 days prescribed, but varies in reality
  - XP: shorter, 2 weeks
- Work in prio order
  - Scrum: team chooses stories in sprint planning
  - XP: team chooses stories strictly according to agreed prio
- Managing change
  - Scrum: no changes in scope / stories during sprint
  - XP: changes allowed

Activity planning

- Product Owner/Customer defines and prioritises User stories
- Backlog or Story card management
- Dependencies are built into the prioritised list, i.e. not explicitly marked

Effort estimation

- Early estimates during exploration phase

- User stories estimated as part of iteration/sprint planning
- Time (man/person hours/days) or relative estimates (story points, bananas, gummy bears)
- Planning game (XP) or Planning poker (Scrum)

Resource allocation

- Capacity driven
  - Project level: In exploration phase
  - Iteration level: In iteration planning
- Within iteration: team pull, i.e. self-allocation

Risk management

- Built into the process, e.g. stand-up meetings: Hindrances?
- Transparency, openess with information
- Iteration demos with customer / product owner

Traditional risk management techniques can be used, but not prescribed by Agile methods as such.

Monitor and control

- Progress monitored by asking ”How much work remains?”
- Frequent status checks -> Burn-down charts, see monitor and control chapter.
- Management does NOT control in agile, the ”pigs” (Product owner, scrum master, team leader) do!
- Team is responsible for managing itself – Team pull!

Team responsible for its own work practice / process

- Regular feedback loops: pair programming, daily stand-up.
- Sprint retrospectives.

Scaling agile

- Designed for small scale – 1 team
- Scalable with Scrum-of-Scrums, shared backlog
- Upscaling specific to each organisation

```
             ----
            | AX |            <-- Scrum of Scrums of Scrums
              / \____..
           __/_
          |ABCD|              <-- Scrum of Scrums
```

```
      _________|_________
   __/_   __/_   _\__   _\__
  |AAAA| |BBBB| |CCCC| |DDDD|   <-- Scrum teams
```

Strengths

- Feedback from early stages used in developing latter stages.
- Shorter development thresholds
- Quickly delivers working increments
- User gets some benefits earlier
- Reduces ‘gold-plating’
- Avoids unnecessary overhead, e.g. keeping docs updated
- Shorten comm paths within cross-functional teams

Weaknesses

- Refactoring causes software breakages-
- Loss of economy of scale.
- Weak / missing documentation – especially for large scale.
    - Dependence on personal knowledge.
    - Decision rationale, reqst-tc tracing may be lost.
- Generalists have weaker specialist competence, e.g. requirements, testing, architecture.
- Weak long-term and overall perspective.

Lecture 4 - resource allocation

In general:

How labor, equipment, space etc. should be distributed to acquire the highest possible efficiency. Performed after activities, effort, and risks have been identified.

- The right amount of people for each job.
- The right amount of work for each employee.

Schedules

- Activity schedules

  Indicating start and completion dates for each activity.

- Resource schedule

  Indicating dates when resources needed and the levels of available resources.

- Cost schedule

  Showing accumulative costs, total spending over time.

Resource schedule and allocation

- Resource types
  - Includes labour, equipment, materials, space, services etc.
  - Time (duration)

    Can often be reduced by adding more staff.

  - Money

    Used to buy other types of resources.

Allocation step by step

1. Identify resources needed and create a resource requirements list.
2. Identify resource types, resources within a group are interchangeble, and resources between groups are not.
3. Allocate resource types to activites, and examine each resource type histogram.

4. If resources needed exceeds the available resources at some time, perform resource smoothing.

5. Resource histograms, where the first is before resource smoothing. Available resources are the dotted line, and the required are shown with lines.

```
Resource
need                   ___
|5|            ___    |   |
|4|....___....|...|...|...|___.....
|3|___|   |___|   |___|   |   |
|2|   |   |   |   |   |   |   |
|1|   |   |   |   |   |   |   |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
           Week
```

```
|5|
|4|....___.....___.....___.___.___.
|3|___|   |___|   |___|   |   |   |
|2|   |   |   |   |   |   |   |   |
|1|   |   |   |   |   |   |   |   |
  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
```

Resource clashes

Occurs when the same resource is needed in more than one place at the same time. Can be resolved by delaying one activity and taking advantage of the float time, or if that is not possible, push back project completion. Could also be resolved by taking resources from a non-critical activity, or by bringing in additional resources - increases cost.

Prioritizing

When prioritizing between two competing activities, there are two ways of doing this.

- Total float priority - Those with the smallest float have the highest priority.
- Ordered list priority - Takes account the duration of an activity, as well as the float, see Burman's.

Burman's priority list

1. Shortest critical activities
2. Other critical activities.
3. Shortest non-critical activities.
4. Non-critical activities with least float.
5. Non-critical activities.

Resource usage

Need to...

- maximise %usage of resources = reduce idle periods between tasks.
- balance costs against early completion date.
- allow for contingency.

```
          /\
Target   /  \   Cost/
scope   /    \  effort
       /      \
      /________\
       lead time
```

Critical chains

When performing resource scheduling, new dependencies may be created, which may lead to additional/new critical chains. Avoid adding dependencies to the activity network to reflect resource constraints. This avoids messiness and a resource constraint may dissappear during the project, but the dependency will remain visually. Instead, amend dates on schedule to reflect resource constraints.

Eventually, actual individuals will have to replace the resource types for tasks. A number of new factors will then have to be considered:

- Availability
- Criticality
- Risk
- Training
- Team building
- Motivation

Cost schedules

Cost schedules can then be produced, these include:

- Staff costs including vacation pay, social security etc.
- Overheads, expenditure that cannot be directly related to individual projects or jobs e.g. office space.
- Usage charges, sometimes projects are charged directly for their use of resources e.g. travel costs or computer time.

Resource allocation concerns

- All schedule resources covered? Specialists, testing teams etc
- All 'heads' are not the same: affects estimates

- Multi-tasking has an overhead time cost
- Overplanning and micro-management
- Underplanning and weak foresight

Monitor & control

Monitor & control is needed since plan != reality. By using this the aim is to reach

- Goals: good product
- Budget: cost
- Timelineness: market window

Monitor

- Objective: check if project is on track with plan
- Different kinds of data/measurements
  - Data from reports
  - Subjective data on completion rate
  - Data comparing actual cost/value and planned cost/value

Cost vs Time

Behind time but ***under budget***, ex: delayed due to not deploying committed staff

On time but ***over budget***, ex: additional resources have been added to cope with work load.

Collecting the data

- Time sheet: Report hours, percentage done, etc.
- Red/amber/green (RAG) reporting: Simply chose the color of likelihood to reach deadline of specific task. Do this for every week/day etc.

Visualizing

- Gantt chart: Basically multiple activity bars, indicating when a task should be started and done, as well as the current progress. Preferably the progress of the bar should be aligned with ***today***.

```
|                    TODAY          |
|--DATE-| 12 | 13 | 14 | 15 | 16 |
Task A:      |====>_________|
Task B: |==============>_________|
Task C: |=============>|
```

Where ***Task A*** is behind schedule, ***Task B*** is on schedule but not done, and ***Task C*** was completed on schedule.

- Slip chart: A Gantt chart with a vertical line showing the progress.

```
|                  TODAY        |
|--DATE-| 12 | 13 | 14 | 15 | 16 |
                   |
                  /
Task A:      |====>_________|
                  \
                   |
                    \
                     \
Task B: |===============>_________|
                     /
                    /
                   |
                    \
                     \
Task C: |==============>|
```

- Brigette's timeline plots planned time along the horizontal axis and elapsed time along the vertical axis. A diagonal line means that the task was delayed, a dot/star means the task is complete.

```
         Task A  Task B Task C
| T | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | .       |       |   |
| 2 |     .   |        \  |
| 3 |         *         \  \
| 4 |             .     |   \
| 5 |                 . |    \
| 6 |                   * .
| 7 |                         .
```

Where *Task A* was finished on schedule, *Task B* was delayed but is finished. *Task C* was delayed even more and is still incomplete.

- Burn-Down Charts (Scrum): A chart displaying the ideal velocity of the project, and a line displaying the actual progress.

```
| T | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | . ___
| 2 |     .\
| 3 |       \ .
| 4 |        \__  .
| 5 |           \_____.___
| 6 |                     .\
| 7 |                       \_.
```

The dots represent the ideal veolocity, the lines the actual.

- Fever chart (critical chain). The upper part indicates that the project most likely will be late, the lower part that the project will finish early, and the middle that it is on schedule.

```
* * * * * * * * * * * * * * * -
* * * * * * * * Y * * *      -
* * * * * * * Y * *        - -
* * * * * * Y Y *        - - -
* * * * * Y Y *  K     - - - -
* * * * * Y    K     - - - - -
* * * * Y Y K K    - - - - - -
* * *   Y K      - - - - - - -
*     K Y      - - - M M - - -
    K   Y    - - M M - - - - -
  K     Y  - - M - - - - - - -
K     Y  - M M - - - - - - - -
  Y Y M M M - - - - - - - - - -
M Y M - - - - - - - - - - - - -
Y Y - - - - - - - - - - - - - -
Y - - - - - - - - - - - - - - -
Y - - - - - - - - - - - - - - -
```

Where *Y* is late, *K* is on schedule, and *M* is early.

Earned value analysis

Earned value analysis is based on assigning a "value" to each task or work, based on the original expenditures forecasts.

- Planned value (PV) or Budgeted cost of work scheduled (BCWS): Original estimate
- Earned value (EV) or Budgeted cost of work performed (BCWP): Work completed this far
- Actual cost (AC) or Actual cost of work performed (ACWS): Actual work performed this far
- Budget at completion (BAC): The budget when the project is done
- Estimate at completion (EAC): Estimated budget at completion, updated as the project progresses

Earned value tracking

When the baseline budget has been established, the project can be tracked as it progresses.

- Schedule variance (SV): EV – AV. Difference between the estimated and actual value.
- Time variance (TV): Difference between planned finish date and actual.
- Cost variance: EV – AC. Indicates the difference between the earned value and the actual cost. A negative CV means that the project is over cost.

Performance ratios

Value for money.

- Cost performance indicator (CPI) = EV / AC
- Schedule performance indicator (SPI) = EV/PV

Example

```
BAC = 100
Actal cost = 80
|===========>        |
    EV = 60 ^        ^ PV = 100
```

- CPI = EV / AC = 60 / 80 = 75 % => over budget
- SPI = EV / PV = 60 / 100 = 60 %
- EAC = BAC / CPI = 100 / 0.75 = 133
- Cost variance = EV - AC = 60 - 80 = -20
- Budget variance = PV - AC = 100 - 80 = 20
- Schedule variance = EV - PV = 60 - 100 = -40

Prioritizing monitoring

Focus on monitoring based on risk

- Critical path activities: Any delay in an activity in the critical path will cause a delay in the project complete date.
- Activities with no free float: A delay of an activity with no free float might cause a delay on some subsequent activities. But it might not delay the whole project.
- Activities with less than a specified float: Same as above.
- High risk activities: Activities that have a high chance of failing/delay.
- Activities using critical resources: Such activities might be expensive.
- Activities with external dependencies: Same

Control

Almost any project will be delayed at some point, the project manager must recognize when this is happening and take immediate action! **dum dum!**

Get back on track

- Try to shorten critical path by adding resources
  - Overtime
  - Re-allocate existing staff to more critical activities
  - Get more staff

    - Reduce quality of some activities
- Reconsider activity dependencies
    - Over-lap activities to avoid waiting for completion of another
    - Split activities to remove dependencies to activities / critical resources

Typical change control process

1. One or more stakeholder might perceive the **need for a change**
2. Affected/receiving party (e.g. customer rep for customer request) decide that the change is valid and worthwhile. **Request for change** (RFC or CR) to the development management for change control management
3. **Change investigated** by developer. Impact and cost estimated. All impact should be considered including testing, long-term maintenance etc
4. The impact investigation is shared with initiating stakeholder who **decides to proceed or not**
5. RFC and its impact discussed with all stakeholders, typically at a **change control board meeting** where RFC is approved or rejected.
6. .... Varied process for implementating change. Should involve **communication** and **tracking of results**

Managing people

Organisational Behaviour Research

Aim: Explain people's behaviour

- Theories á la "If A then B is likely to follow"
- Dependencies, not causal relationships & formulas such as e.g. Chemistry, Physics

Taylorism

Select the best people for the job (**!**), instruct them in the best methods, performance-related pay.

The problems with taylorism is the inflation of the performance-related pay, and staff exhaustion.

Hawthorne effect

By showing an interest in a group the group would perform a lot better, even though there were no change in work.

Theory X / Theory Y by McGregor

Theory X

- The average human has an innate dislike of work
- There is a need therefore for coercion, direction, and control
- People tend to avoid responsibility

Theory Y

- Work is as natural as rest or play
- External control and coercion is not necessary
- Commitment and objectives is a function of reward
- The average human can learn to accept and further seek responsibility
- The capacity to exercise imagination and creative qualities distributed

Herzberg

- Hygiene or maintenance factors make you dissatisfied if they are not right, e.g. pay or working conditions
- Motivators make you feel the job is worthwhile, e.g. a sense of achievement or the challenge of the work itself

The Oldham-Hackman model

The satisfaction of a job is based on five factors, listed below. The first three factors (skill variety, task identity and task significance) make the job "meaningful" to the person who is doing it.

1. Skill variety: The number of skills used in the exercise
2. Task identiy: A sense of ownership of your work
3. Task significance: Influence on others
4. Autonomy: Responsibility, authority. Discretion about the way that you do the job
5. Feedback: Information you get back about the results of your work

According to Oldman and Hackman, activities should be designed so that staff follow the progress of a particular product and feel personally associated with it.

Stress

Stress is reduced with good project management. Manage the resources properly (no overtime), clear goals, proper monitor and control to avoid a crisis.

Working in teams

Project roles & responsibilities

Project sponsor or director
: Secure budget + resources. Champion of project goals, ultimate decision maker. Approve changes, progress, sign-off deliverables etc.

Steering committee
: Resp for overall project success. Representatives for key stakeholders and involved organisational units. Support project sponsor by providing multiple perspectives.

Project manager
: Day-to-day resp to ensure project meets goals (Scope-Time-Cost) by planning and managing project and project team. Secure acceptance of deliverables from sponsor and stakeholders.

Stakeholders: Key and other
: All project roles and others who may be impacted by the outcome of the project.

Customer respresentatives/decision-makers
: Active and available to project on matters of customer interests, e.g. regarding requirements and acceptance of deliveriables.

Vendors
: Contracted to deliver products or services to the project.

Project team members
: Execute tasks, produce deliverables as directed by project manager.

Organizational structures

An organizational structure is needed to form and manage groups in large projects.

Formal versus informal

The ***formal*** structure focuses on ***authority***, about who has which boss. This is usually the base of the organization's structure. When the "unexpected happens" the ***informal*** structure takes over, this structure consists of spontaneous contact and communication between the members of the staff while working. The informal organization gets built up and unofficial ways are found around the obstacles imposed by the formal structure.

Hierarchical approach

Each member of the staff only has one manager, and the manager has responsibility for several staff members. Authority flows top down. A concern with this approach is ***span of control*** - the number of people that a manager control.

Staff versus line

Line workers produce the end product, support ***staff*** carry out supporting roles (think IT-department).

Reporting

Formal

Regular

- **Oral** Progress meetings
- **Written** Job sheets

Ad Hoc

- **Oral** Review meetings
- **Written** Issues reports

Informal

Regular

- **Oral** Around the coffee machine
- **Written** Emails to collegues

Ad Hoc

- **Oral** Ad hoc meetings
- **Written** Emails for issues investigation

Leadership

Leadership is the ability to influence other and achieve a common goal. Management includes leadership, as well as organising, planning and controlling. Both exercise power but in different ways, position power (formal authority; rewards, punishment) or personal power (individual qualities; an expert, access to information).

Programme & Portfolio Management

Project: Business case & contracts

A business case may be presented for several potential projects. It should show that the benefits of the project will exceed development, implementation and operational costs. This is part of portfolio management.

Content of a business case:

Introduction/background
: describes a problem to be solved or an oppertunity to be exploited.

The proposed project
: a brief outline of the project scope.

The market
: the project could be to develop a new product. The likely ***demand for the product*** is assessed.

Organizational/Operational infrastructure
: describes how the structure of the organization will be afftected by the implementation of the project.

Benefits
: These should be expressed in financial terms.

Outline implementation plan
: how the project is going to be implemented.

Costs
: the implementation plan will supply information to establish these.

Financial plan
: combines costs and benefit data to establish value of project.

Risks
: initial risk analysis to identify risks to project execution and to business aspects that affect profit.

Management Plan
: Missing information but part of the business plan.

Selection of project approach

steps from Step Wise chart

- **2 Organization structure needed**
  - Make/buy/outsource
  - Make/reuse/
  - Make/share

  - ...
- **3 Process model**

Buying from external suppliers

Different buy-situations:

- **Bespoke system/component**
- **Outsourcing a task**
- **Commercial off-the-shelf (COTS)**
  - bought *as is*
  - customized

Types of contract

- **Fixed price contracts**: fixed price, terms, requirements & delivery time
  - - known cost, supplier motivation
  - - higher price to absorb the additional risk, frozen requirements, ***impossible*** cost -- quality risk
- **Time and materails contracts**: fixed rate/unit of effort
  - - requirements changes, lack of price pressure
  - - customer liability due to uncertain cost & commitment, lack of incentive for supplier
- **Fixed price per delivered unit**: incremental delivery and payment => a series of contracts
  - - customer understanding price calculation, comparability between pricing schedules, emerging functionality can be accounted, supplier motivated to be cost-effective
  - - difficulties with SW size measurement - may need independent fixed price counter, changing requirements - how do you charge?

Cost-benefit analysis (CBA)

- **Identify all the costs which could be:**
  - development costs incl buy-ins
  - set-up incl recruitment and training
  - operational costs after set-up
- **Identify the value of benefits**
- **Check if benefits are greater than costs**

Costs and benefits must be identified and expressed in the same unit..

Liftime of Return on Investment (ROI) for potential project

- **Estimate timing costs and income**
- **Development incurs costs**
- **Release of system/product generates income - gradual pay off**
- **Include decommissioning cost**

Typical project life cycle cash flow

```
  |
i |
n |
c |                              *      *
o |                        *                     *
m |                     *                                *
e |                   *                                           *
------------------------*----------------------------------------------------------
*------*-
e |*                *                                            time -> *    *
x | *              *                                                       *
p |  *            *
e |   *          *
n |     *       *
d |        *  *
i |
t |
u |
r |
e |
```

Cost-benefit evaluation techniques

- **Return on investment (ROI) or also known as Accounting Rate of Return (ARR) = Net profit / total investment.**

  Also useful for company individual choices or features.

- **Net profit:**
  - total income - total cost
- **Payback period:**
  - time to break even

```
      average annual profit
ROI = ------------------------ X 100
        total investment

      value in year t
NPV = -----------------
         (1 + r)^t
```

- **Net present-value**

        - Present value of future cash flow
    - **Internal rate of return (IRR)**
        - Internal rate of return (IRR) is the discount rate that would produce an NPV of 0 for the project
        - Can be used to compare different investment opportunities
        - There is e.g. a Microsoft Excel function which can be used to calculate the IRR (or use a suitable technique from numerical analysis)

Portfolio Management (PM)

Porfolio project management provides an overview of all the projects that an organisation is undertaking or is considering. The optimization coverage of product ranges and market segment are need & profit.

- **Predicting market**
- **Evaluating proposals for projects**
- **Checking for gaps & overlaps**

For potential projects

- **Balance total risk**
- **Resource sharing**
- **Dependencies between projects**

Probability of

```
               success
             high ^
                  |
                  |
                  |
    xxx           |               x
  xxxxxxx         |              xxx
    xxx           |               x
----------------------------------------------> Profit
low               |        xxxxxx   high
          xx      |       xxxxxxxx
        xxxxxx    |        xxxxxx
          xx      |
                  |
             low  |
```

Optimize Over Time

```
-------------------
|  Project 1     |
-------------------
```

```
------------------------------------------------------------------------------
|  Project 2                                                          |
------------------------------------------------------------------------------

                -------------------------------------
                |   Project 3                    |
                -------------------------------------

                                      ---------------------------------------------
                                      |   Project 4                          |
                                      ---------------------------------------------
-----------------------------------------------------------------------------------
------>
                                                                        time t
```

Issues with PM

- Will another project be more profitable?
- Hard to express benefits in financial terms
- Projects with the highest potential returns are often the most risky

Programme Management

One definition:
*a group of projects that are managed in a **co-ordinated** way to **gain benefits** that would not be possible where the projects to be managed independently*

- Business cycle programmes - common time & resources
- Strategic - with common goal
- Infrastructure programmes - e.g. IT support projects
- Research & developemt (R&D) development programmes - *safe* + *risky* projects
- Innovative partnerships - new technologies, precompetitive

Responsibilities of a program manager

```
-------------------------------------------------
|                                                |
|   Stakeholders, sponsors                       |
|________________________________________________|
                      /\
                     /  \          |*Vision
                      ||           |*Mandate
                 _____||_____      |*Cost estimates of projects
                 |  Program |     / *Risk for projects
                 |  Manager |     \ *Resources
                 -----||-----      |* ...
                      ||           |
                     \  /          |
                      \/
-------------------------------------------------------
|                                              |
|   Projects                                   |
|______________________________________________|
```

Project Management at Programme Level

```
High Level activity plan
```

```
  B                                            G
  ---------------                                ---------------
  |Corporate    |_____________________________\ |Impl. corp   |
  |image design|                 /|\         / |interface    |
  ---------------                   |            ---------------
                                  |
A                 C               |            F
---------------   ---------------    |            ---------------
|System study|___\|Build common|__|___________\|Data         |
|/design     | | /|systems     |  |          /|migration    |
---------------  |  ---------------    |            ---------------
               |  D              |   E               /|\
               |  ---------------  |   ---------------    |
               |_\|Relocate    |_\|/_\|Training    |___|
                 /|offices     |    /|            |
                  ---------------      --------------
```

To this activity plan, an Gantt chart is made to overview the plan.

Quality Management

Quality is about meeting the minimum standard required to satisfy customer needs.

ISO 9126

Quality in use

- Effectiveness – ability to achieve user goals with accuracy and completeness
- Productivity – avoids excessive use of resources in achieving user goals
- Safety – within reasonable levels of risk of harm to people, business, software, property, environment etc
- Satisfaction – happy users!

External Software Quality Characteristics

FEMURP

- **F** unctionality – does it satisfy user needs?
- **E** fficiency – relates to the physical resources used during execution
- **M** aintainability – relates to the effort needed to make changes to the software
- **U** sability – how easy is it to use?
- **R** eliability – can the software maintain its level of performance?
- **P** ortability – how easy can it be moved to a new environment?

TQM - Total Quality Management

```
 __________       ______________
|          |     |              |
| Focus on |     |   Improve    |
| processes|_____|_continuously |
|_________/        \____________|
         /          \
        /  Focus on  \
        \  customers /
 ________\          /______________
|         \________/               |
| Base decisions | | Let everybody |
|    on facts    | | be committed  |
|________________| |_______________|
```

> ```
> Committed leadership
> ```

Practical

- Quality plan
- Define criteria for activities & include inspections
- When are we done?
    - Monitoring quality levels
    - Release decision based on quality measurements

Quality Plan

A quality plan can be seen as a checklit that all quality issues have been dealt with by the planning process.

- scope of plan & references to other documents
- quality management, incl organization, tasks, and responsibilities
- documentation to be produced
- standards, practices and conventions, reviews and audits
- testing strategy and plan
- problem reporting and corrective action
- tools, techniques, and methodologies
- code, media and supplier control
- records collection, maintenance and retention
- training
- risk management - methods of risk management that are to be used

Clear criterias for activities/inspections

- Entry requirements - test data and expected results prepared
- Implementation requirements - "when error is found, do this and that"
- Exit requirements - done when all tests ok

When are we done?

It is not possible to **know** when there are no more errors, this has to be estimated. Using testing techniques or expert opinion.

Result

- Successful?
    - Delivered on time
    - Delivered within budget
- Gains
    - Quality product that meets customer need
    - New lessons learnt (experience, process improvements)

Summary

- Quality is a vague concept, things have to be carefully defined
- Most qualities that are apparent to users can only be tested with the complete product
- Need ways of testing during development to see how the final product will behave
- Some techniques concentrate on testing the product of the development process, while others test the process itself.