Elements of IT Project Management

Script accompanying my lecture

University of Applied Sciences
Frankfurt am Main

June 2015

Dr. Erwin Hoffmann

Version history:

- First Version:
  The first version was created upon my first lecture about IT Project Management in 2008. It included most of the material available today and referenced the "inclined tower of Cologne" as sample of bad Project Management. The layout was realized by means of *Ami Pro* under *Windows 2000*.
- Second Version:
  This version results on my 2nd lecture about IT PM in 2009 improved on Project Management regarding the German DIN standards. Due to current events, I extended the problems with bad Project Management with the collapse of the historical municipal museum in Cologne. This script was realized with *OpenOffice* under *MacOS X*.
- Third Version:
  Here, additional 'Agile Project Management' is included. Unfortunately, I had to extend the chapter about bad Project Management in Cologne even further (2010). If anybody believes, Project Management is not necessary, the showcase is open. The script was completely revised, ported to *TeX* and realized with *Texmaker* under *Ubuntu* and fully hyperlinked.
- Forth Version:
  Included chapters about *Scrum*, *6σ*, *User Centered Design*, and *Continuous Integration*, extended the *Quality* section and re-organized the script. Epigraphs for the 'part' pages give me headache. Using now *TeX* on *MacOS X*.

Contents

Part I

Introduction

Building the Pyramids took a lot more planning than Linux.

Linus Torvalds (2012)

Chapter 1

What is Project Management?

1.1 Scope of the course

This course summarises my experience of now almost 10 years in Project Management (PM) and provides an outline of today's established methods and standards used to manage IT projects successfully. While in particular two distinct frameworks for Project Management exist (PMI and PRINCE2), I rather will focus on practical issues which are in particular specific for IT projects.

The course is organized in five parts:

- Part one shall provide an overview of IT-PM.
- Part two deals with the fact, that IT projects are typically set up in an existing landscape and PM has to consider this heritage which can be beneficially but occasionally might be a heavy burden a new project.
- Part three considers Team Management as probably the most import factor for a project success or failure.
- Part four lays out some principal ideas of project planning.
- Part five provides an overview of general Project Management methods and existing frameworks.
- Part five, finally, is dedicated to the specifics of IT projects, dealing in particular with *Software development* while including Software Modelling, Quality and Defect Management as well as Integration and Release Management.

1.2 Historical Projects

1.2.1 Building the Cheops Pyramid

By today's knowledge, the Cheops pyramid at Gizeh (one of the Seven World-wonder) has been started to get build 2467 BC (according to stellar constellations). It should serve as tomb for the pharaoh Cufu, which gave order to

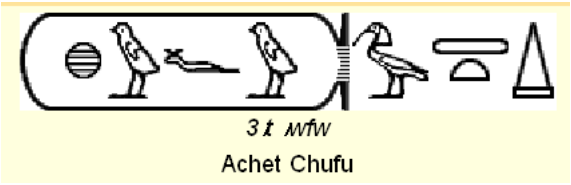

Figure 1.1: a) The Cheops Pyramid in Gizeh; b) Achet Chufu = Horizon of Chufu (Cheops) [49]

build the 'big pyramid' shortly after his father Snofru died [1.1(a),1.1(b)]. It is believed, that the building was completed 30 years after this order was received, and it is estimated that about 10 years were used to establish the (absolute plain) platform and 20 years were enough to complete the whole building. Not only that the tiptop is missing today (10 m) but in addition the big pyramid was originally equipped with additional tura lime stones painted snow white and stood as tallest member among the *Chephren* and the *Mykernios* pyramid in the valley of *Gizeh*.

During those 30 years, not only the architecture of the pyramid had to be designed, but in addition the necessary material and the labour force had to be organized and sustained over this period. 2,5 million limestones had to be broken downside the Nile and transported (by ship) to the building place, each of them had a weight of 2.5 tons (in average) to be finally lifted up to an altitude of 150 m. Probably, even with today technologies, we would no be able to complete such a building (in the required quality standard) in that time. Without an almost perfect Project Management, this task would be never achievable.

We have to consider that not only all the material had to be transported here, but also special tools and wooden and perhaps copper lever arms needed to be constructed. Several thousand partially high skilled people had to be

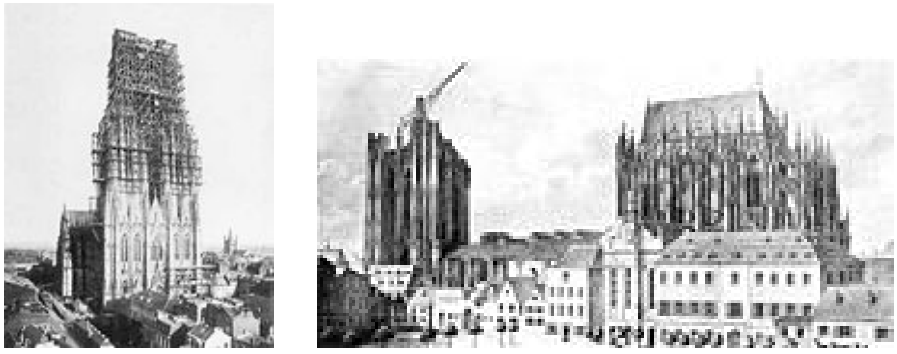

Figure 1.2: Cathedral of Cologne before (left) and after (right) finishing the South Tower [53]

fed; they needed hosting as well. Since those felachs typically were needed for farming, additional care had to be taken that they are substituted and that the overall economy is still working.

1.2.2 Building the Cathedral of Cologne

The Cathedral of Cologne was planned to be build in the year 1248 in order to serve as tomb for the remnants (*reliques*) of the 'Holy Three Kings' (visiting the Jesus boy child shortly after his birth). The trachyt stones are gathered from the nearby 'Siebengebirge' down the river Rhine and shipped to the building place. The first part of the 146 m tall building was finished in 1322.

However, constructing the South Tower was not completed until 1880 while using the original plans (figure [1.2]).

Due to environmental influences, the trachyt stones age significantly and the Cathedral needs permanent repair and fixing. Building of the Cathedral of Cologne will probably never stop in order to keep this building as 'UNESCO world heritage'. A common phrase of the citizens of Cologne is: "Once the Cathedral is finished, the world is at it's end." I am not talking about Project Management here; guess why.

1.3 What is a Project?

Probably the first modern (and actually written down) definition for a 'project' originates from R.L. Martino [26]:

> "A project is any task which has a definable beginning and definable end and requires the expenditure of one or more resources in each of the separate but interrelated and interdependent activities which must be completed to achieve the objectives for which the task was instituted."

DIN 69 901

In Germany , the 'Deutsche Industrie Norm' `DIN 69 901` provides in addition a precise definition of the term 'project' [4]:

> "A project is task defined by the uniqueness of it's all-over realization conditions, in particular a
> (1) defined aim or goal,
> (2) defined conditions regarding time-frame of realization, financial, personal, and other conditions,
> (3) is distinct against other tasks, and
> (4) provides a project-specific organization."

Generally speaking, within a given organization (whether a company, GO[1], or even among your family, colleagues and friends) a 'project' is understood as the opposite of 'general operation', thus it is something special and requires particular attention and funding. In our today's understanding, a 'project' neither requires a certain complexity, nor is a 'project' characterized by virtue of its 'technical' character. As introduced, even building a family house or buying a family home might be a project.

On the other side, our common notion of a project would include the attributes 'challenging' and 'complex'. According to [24] a project can be characterized as:

- Limited task with defined start and end (goal)
- New horizons: Touching the limits of current technologies
- Risky in terms of technology, economical impact, and time-frame
- Complexity:
  - Lots of participants of different disciplines; perhaps third party organizations
  - Interdependencies not standardized; organizational structure not (yet) established
- Moving organizational requirements during the project phases
- Substantial impact for the company or organization
- Time to market dependencies

Figure [1.3] lists (on a time scale) some 'modern' projects where Project Management methods were used. Of course, the complementing picture ('how many project failed though PM was tried') is much more difficult to achieve. In addition, the quality of the solution achieved has to be questioned. However, this is outside the scope of Project Management and in practice a hot discussed political issue.

[1] Government Organizations

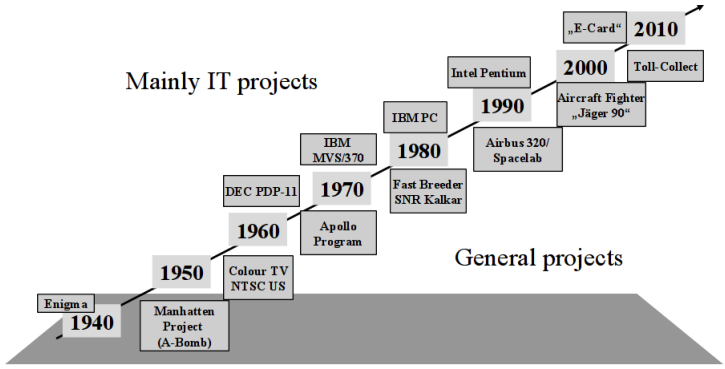

Figure 1.3: Projects managed with Project Management support [24]

1.3.1 What is Management?

Let's try to define what management is:

> "Management includes all necessary tasks to plan, organize and supervise the current activities based on explicitly expressed methods, thus the project is eventually completed within time and budget and the product (the result of the project) can be achieved with determinable functionality and quality."

Management

As a result, (project) management has to balance the goals and the effort which we will later will call the 'magical triangle of project management'.

Here, we have mentioned the management elements

- planning,
- leading, and
- controlling.

During the project's progress these importance of those elements will undergo substantial changes. While in the begin of a project, clearly the planning of the product and the project has preference, during the operational phase leading and controlling deserves most attention.

Often, the latter tasks are split among different people or even organizational units. As we know from many projects, this view is inherently incomplete. In particular large projects need governance:

Governance subsumes the infrastructural and organizational means to provide during the lifetime of the project the required framework to ensure

Governance

- dedicated technical competence,
- operational responsibility, and
- overall risk management.

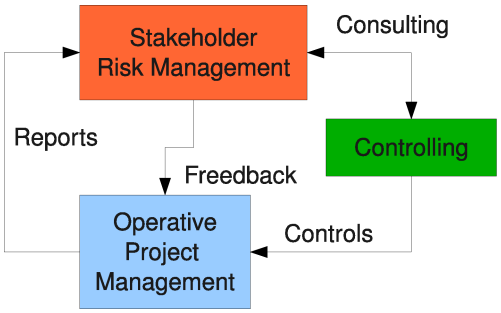

Figure 1.4: Framework for Project Governance setup

Thus governance is mostly situated outside the project: it serves as an umbrella. However, governance is a joint relation between the responsible project's stakeholder, the operative management, and the project controlling to commonly share the *estimate-able* and *none-estimate-able* (residual) risk of the project and provide a framework for effective risk management (figure [1.4]).

1.3.2 Why Project Management?

There is a general understanding, that complex tasks needs some special organization in order to succeed and meet the project goals. The driving factors for a dedicated PM are:

- Reduced 'time to market' for new products.
- Estimate the amount of resources (budget, people, and other conditions) needed.
- Control the project in every phase and allow a reasonable risk management.

Project Management can be characterized as target disc. Main task is to meet the project goals, while the surrounding 'rings' may serve to hit this target; though badly organized can lead to substantial deviation (figure [1.5]).

1.3.3 What are failed Projects?

Failures

Typically, projects fail if more than one out of the following conditions are met:

- The project runs out of scope
  The original aim of the project can not be reached. This happens, if no realistic business plan was established,
- The project runs out of business
  The market conditions have been changed substantially during the running project, thus it became more or less obsolete.

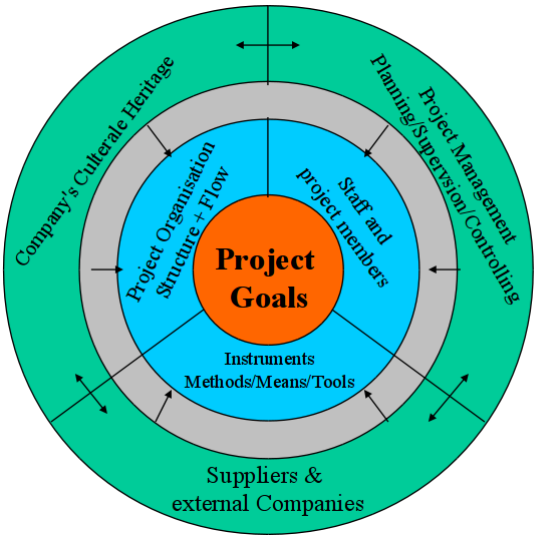

Figure 1.5: Elements of Project Management [24]

- The project runs out of time
  This may happen, if the complexity of the project was underestimated, or the funding was insufficient, or people "dilute" from the project.
- The project runs out of budget
  Sadly to say, that most of the projects have to suffer from that fate. One of the last prominent projects to mention is the German *lorry maut* system finally realized by *Daimler-Chrysler*[2] and *T-Systems* (called the *T-Collect consortium*), though it was a successful project regarding all other issues.

By experience, not all projects can be successfully completed. For instance, it was common practice of *Microsoft* (and naming a German company: *Softlab*) to set-up up different project teams for the same project running in competition. One goal for a project might also be to build strong teams, capable to successfully work on the next project. However, under today's economical pressure, this policy becomes more and more uncommon. By the very same token, quite often projects are announced 'successfully completed' and 'ready for operation' if in reality the system is only partially working and does not meet the predicted (essential and/or important) project goals.

1.3.4 Project Commitments and De-commitments

In order to partially complete a project successfully (*in time and in budget*) it might be necessary to reduce the original stated aims such they fit into

[2] No *Chrysler* anymore!

the current situation of the project. This situation may become apparent at virtually any phase of the project.

In case the project depends on a third party delivery, the supplier has to actually *de-commit* parts of the announced functionalities. Of course, this requires to change the original project plans in order to re-evaluate the dependencies and estimate the consequences for the whole project. In case of a substantial impact, a common procedure is to have already considered alternative solutions (*plan 'B'*), i.e. having a different potential supplier in 'stand-by' mode.

Criterion's

It is important to be able to distinguish between

- 'essential' (absolutely necessary),
- 'important' (but substitutable) and
- 'nice to have' (can be left out)

features of the project.

For all important and in particular essential goals of the project initially, a so-called *'critical path analysis'* has to be carried out as part of the *risk management*.

Background, Exercises, Facts

A known disaster of Project Management: *The Cologne Subway Drilling*

While I prepared this lecture and was referencing the building of the *Cathedral of Cologne* I (and many others witness with me) a complete failure of Project Management.

In Cologne a new underground line is going to get build, touching the Cathedral closely and needs extensive drilling. In 2007, due to the same construction, the tower of the *St.-Johann-Baptist* church was heavily impacted [1.6] and needed some 'resurrection'. Disaster #1

At the same time, it became clear that some static calculations about Cologne's subterranean structure and capabilities is obviously odd and need re-calculation. However, the incident was considered 'single cause' only and consequences were neglected.

Figure 1.6: St.-Johann-Bapist Church victim of today's (none existing) Project Management [Der BauUnternehmer]

One year later, in Spring 2008 and less then 1 kilometer further south (location 'Waidmarkt') the *Cologne's historical Archive* collapsed out the the nothing [1.7] Later it was identified, that the companies in charge of the civil construction pumped much to much ground water into the river Rhine. Thus the underground – consisting of sand and banked up by the river over thousands of years – simply 'diluted' while filling up those areas where the ground water has been sucked away from. It was identified hat to many ground water pumps had been used, far above what was originally allowed. Sadly two people died (fortunately only) the costs of the projects raised from 600 mio Euro to 1.2 billion Euro. The estimated losses for the historical building can only be estimated to be at least some further 500 mio Euro. Disaster #2

Again one year later (during the carnival season 2009) it became obvious, that roughly about 85 % of the supporting metal frames were stolen by some workers at the construction site [1.8]; in particular obvious at the 'Alter Disaster #3

Figure 1.7: Collapse of Cologne's Municipal Archive building in March 2009 due to civil construction (drilling) for the new subway line [32]

Markt', one of the most important spots for the carnival rally.

Figure 1.8: Civil construction site at the 'Heumarkt' where 85 % of the supporting metal frames where stolen by local workers [36]

According to the news agency WDR [36]:

> KVB [the local public transport company] lawyer Gero Walter declared (08.03.10) that the civil construction controls at the site 'Waidmarkt' did not work. Though he emphasized that the cause of the collapse of the Municipal Archive Museum has not been determined yet. Prior of the collapse the workers did essentially control themselves. An independent engineer simply checked the static calculations and protocols available to him. However, he never visited any construction site, which was common otherwise. After the collapse, happening on March 3rd 2009 , the regional government at the capital Düsseldorf detracted the responsibility for the construction controlling from the KVB. Since then (!) the local constructions are additionally supervised by an external consulting company.

Chapter 2

What is IT Project Management ?

IT Project Management is part of the general PM, however due to the nature of the delivered product the software special tools and frameworks can be used to guide, manage, and measure the quality of the product and finally to deliver the product in a well-suited form ready to use in an easy way.

IT Project Management is a part of the general PM and has *per se* neither less nor more management requirements as any other project; though of course specific. However, in the IT industry and business it is often believed, that IT project management can be facilitated with IT means and tools only. This is a substantial mistake which yields finally to the failure of the project. In particular, there are some substantial "don'ts" in IT PM:

The big "don'ts"

- *General Management by Powerpoint (or equivalent)*
  this is a bad habit, and often staff and project members are fed reading those slides.
  It is believed, that the fate of the *Space Shuttle* '*Columbia*' and the death of all astronauts, evaporating in the earth' atmosphere, due to fallen-off heat-tiles during the space craft's lift-off are due to an miss-interpreted *Power Point* slide [14].
- *Document Management by Word (or equivalent)*
  any complex document structure to be consistently maintained needs a special system; systems based on binary representation simply don't work.
- *Project Management by Outlook (or equivalent)*
  neither communication nor deadline management should use extensively email communication; email by construction is unreliable.
- *Problem/Incident/Defect Management by Excel (or equivalent)*
  these tools are good for reporting numbers, but incapable to allow a content-driven analysis.

IT Project Management consists verbally of three different terms in reverse order: *Management*, *Project*, and *IT*.

Civilization

Management is part of the human society and culture. Actually even animals manage their inter-relations and their common search for nutrition. Ants, bees, dolphins, zebras, all these species are highly organized and partly well managed. Management results as part of Darwin's law to react upon external circumstances and to optimize their behaviour in order to survive as species. Management can be described in this sense as leadership (active management) and common, instinctive behaviour (self management or self organization). Human 'management' is not free from those elements, but we belief that cultural and civil progress is based on the fact that human beings substitute those elements with intellectual leadership and rational self-organization, though the history of the last hundred years seems sometime to be in opposite of this opinion [34].

Organization

Project (management) in today's understanding is an organized way to approach a *pro-jection*. Probably since the first scientific articles on Project Management were written down [26], it became evident, that concerning the finite duration of a projects, it's evolution can be characterized in terms of life-cycle or project-phase models. How to describe those phases in detail and how to deal with them is part of different Project Management schools.

Information Technology

The third basic term *IT* is actually not *Internet Technology* but rather *Information Technology*. Information Technology is based on a *Computer* as a hardware device and a *Program* specifically build for that Computer, which we reference as *software*. Up to the 70th, Hardware and Software was originally bundled, thus the software and in particular the Operating System (OS) of the hardware was not particular offered and purchased but rather part of the Computer system itself and development. New releases were covered by the license fee and maintenance contract required to run the Computer system. Of course, at that time dozens of Computer suppliers were in competition. Apart from the big players *IBM* and *DEC*, still *Honeywell*, *Interdata*, *CDC*, *Nixdorf*, and *Wang* had their substantial share. All of those were supplying completely different and vastly incompatible hard and software solutions: From connectors, cables, machine representation of bytes and words, system architecture, programming languages, communication stacks, tape and disk drives, terminals, and of course operating system and the human interface.

While those companies made a lot of money at these (gold rush) days, the situation for the customers and users were rather unfavourable. In particular the scientific communities required a common computing framework using a specific high-level program language to be generally used (FORTRAN) and developing sets of libraries for scientific calculations.

Further, inter-communication between different computer systems was still difficult. Data transfer was facilitated by tapes (reels) written in a standard format, program-to-program communication among the vendors still

unheard of. While the **OSI**[1] tried to develop a common computer communication framework, the **DoD**[2] has already succeeded setting up the so-called **ARPANet**[3] in the US, running the predecessors of the TCP/IP protocol suite. This progress was partly based on a new mid-level computer language developed by *Kernighan and Ritchie* at the Bell Laboratories originally implemented on UNIX Operating System (OS) running on a DEC PDP-11 [23]. The UNIX OS already provided the TCP/IP communication stack in the *C language* and due to the unrestricted license policy by the Regents of the University of California, UNIX became wide-spread in the academic world [38].

While these developments took place in the scientific community, the IT business regarding commercial solutions was still in the hand of the hardware vendors. This changed substantially with the advent of the Personal Computer (PC) by IBM in 1981. Now a standardized platform was introduced which allowed to decouple hardware progress from software development. Start-up companies like *Novell*, *Microsoft*, *Lotus*, *Corel* and many others used this new platform to offer customer and end-user specific software products.

UNIX

For today's understanding of IT Project Management the elements UNIX with it's scientific background standardization of the OS (not to forget **POSIX**[4] DIN/EN/ISO/IEC 9945 compliance), reduction of computer languages and hardware, and of course the rapid introduction of the Internet are all equally important.

Practically, no IT project can be set-up 'confined' to service one platform only, but rather has to interact with lots of external systems by means of (more or less) standardized interfaces. Naturally, this yields a certain amount of complexity to IT projects and requires specific management skills and tools.

2.1 Tasks and responsibilities of a Project Manager

Duties and responsibilities of a project manager or a *Project Leader* (**PL**) depend of course on the tasks upper management has assigned to him or her. Further, since PLs may change during the evolution of the project, the tasks depend on the state of the project. It is not uncommon to change *Sub-Project Leaders* (**SPL**) under certain conditions, in particular if those conditions become critical or even over-critical.

Questionnaire

Starting as a **PL** , the first and most important task is, to ask questions. A possible breakdown of questions can be found in figure [2.1] [33].

Experiences

The second most important aspect is, whether you as a **PL** have already

[1] Open System Interconnect
[2] US Department of Defense
[3] Advanced Research Project Agency Network
[4] Portable Operating System Interface

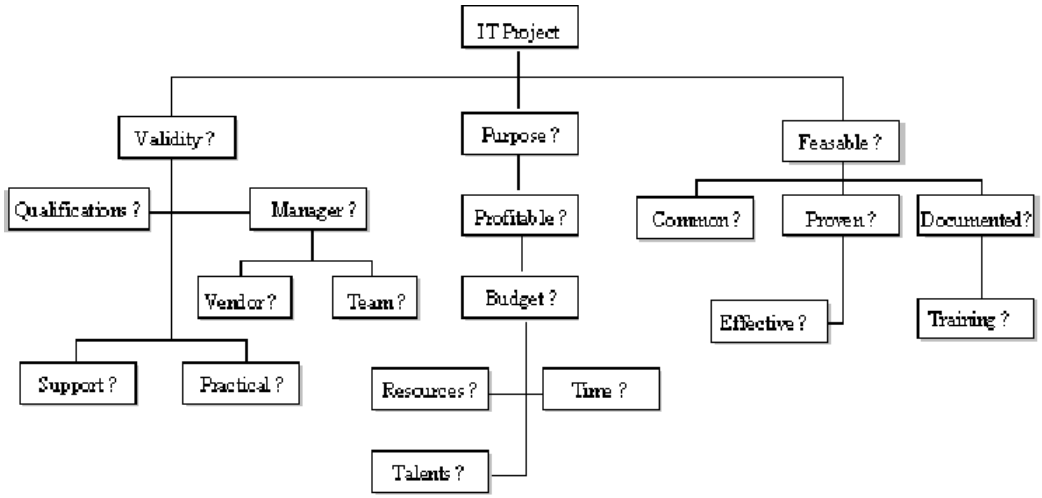

Figure 2.1: Project Manager's Questionnaire to upper Management [33]

experience in the technical field described and/or any project management experiences. Both circumstances have to be clearly signalled to upper management. Of course, knowledge can be gathered in the running project; but has to be present at critical phases. The third but in practice often neglected item is, whether you are capable to personally guide and manage a team of people. This requires involvement (in particular emotional), communication skills, trust in your own work (to be signalled to the project team), and certainly a large amount of standing, in particular against upper management.

2.2 What are Stakeholders ?

Today's IT projects are typically not confined. While back in the 90s, *Sun Microsystems* was able to develop and promote the JAVA computer language practically from scratch, it is more likely that even any new idea has to be placed in the real existing world.

Whether a new software has to be introduced in your company, or you plan to roll-out a new product 'compatibility' is a key-factor for success. In order to additionally achieve economical success, 'acceptance' is another buzzword the project should aim for. While the first is an technical issue, to achieve 'acceptance' requires human intervention and explanation to people, whether internally, business partners, or potential customers.

Stakeholders

In general , all parties which have an interest on the project should be considered as *Stakeholders* and consulted frequently. Stakeholders are in return important for the final success of the project and can act as friends or foes in particular during critical phases of the project. Thus, it is import

not only to report the progress (and/or problems) to those, but also to take their concerns serious and to react on those. Certainly, it is a good idea to document the project's progress and phases transparently, i.e.. on an Intranet Web site.

According to [33] stakeholders include:

- (Upper) Management
- The project manager (PL)
- The project team
- Project sponsors
- Customers
- End users
- The community

2.3 Project phases

According to our current understanding, any project needs to be subdivided into four phases:

1. Initiation – get the idea, study feasibility, define a raw project layout — Initiation
2. Planning – detail project including schedule and resource planning — Planning
3. Execution – final detail product design, coding, testing, implementation — Execution
4. Termination – release and roll-out of the product, terminating the project, lessons learned — Termination

The duration of each individual phase depends on the *complexity of the product*, as *subject of the project*. *How* we realize the planning, and *how* we organize the execution phase depends on specific project management methods used. — Product = Subject of the Project

In particular for larger projects, the *planning phase* could be organised as project in itself. The coding and testing of the software during the *execution phase* is often delegated to external companies which need to setup a project plan by them self. Thus, we talk about a *Multiproject Planning* or a *Programme*. — Programme

2.3.1 Project versus Product

Identifying the *Product*, to be *Subject of the Project*, we need to consider two interrelated streams:

- Project planing, execution, management and supervision

- Product design, realisation, while conforming to specification and quality

However, running a *project* is always determined by realizing the *product*; the project's goal.

Often, the principals of of project management are tailored to achieve this aim, or even sacrificed in case the current methods seem to be inappropriate to realize the tasks.

Product development = Driving factor for the Project

The product thus has a significant impact on the project: Product development is driving the project. After more than 50 years of SW development, we understand more deeply, that *Software* as a product has it's own merits determining the project's organization, which yield a typical and distinguish *IT Project Management*.

2.4 Project Management: Compliance and Conformance

Today's IT Project Management is based on a stet of standards:

- General Project Management Frameworks, including in particular methods for Quality Management
- Software Engineering Models, including SW design, integration, (automatic) testing, while using special Frameworks and Tools for Source Control and Defect Management

The most global and uses project management frameworks and standards are now briefly introduced.

2.4.1 Project Management Institute (PMI)

PMBoK

Project Management Institute (**PMI**) founded in 1969 published in *Project Management Journal* 1983 a special report containing the results of the so-called **ESA** project. This is considered to the predecessor of the *Project Management Body of Knowledge* [21] and used as a framework for **PMI** in order to set-up a program for further accreditation and certification.

The PMBoK is a *knowledge based* Project Management approach and acts as one of today's PM references and is filed as ANSI standard `ANSI/PMI 99-001-2004`.

2.4.2 PRINCE2

PRINCE2 and ITIL

PRINCE2 is the synonym for '*PRojects IN Controlled Environments*', Version 2. It has been developed by the British '*Office of Government Commerce*' **OGC** [30] which is also founder of **ITIL** (*IT Infrastructure Library*) [29] recommendations. While it is in use since almost 25 years now, unlike

PMI it follows a more easy and *process-oriented project management approach* known as '*Management by Exception*'. Thus, it provides a framework of 'what to do' in specific situations, unlike **PMI**'s "how to do". PRINCE2 can be adopted to any projects not just for IT projects and is widely used in industry.

Potential project managers can be certified by accredited organizations Comparable with ITIL, the first degree is 'Foundation' while the more advanced user may achieve a 'Practitioner'.

2.4.3 ISO 10006 / DIN 69 90x

The German **DIN** ('*Deutsche Industrie Norm*') `69901` is the predecessor of the `DIN 69900` which introduced the so-called '*Netzplantechnik*' and defines in it's first part the basic terminology for project management. The third part however, introduces a *work breakdown structure* (**WBS**; german '*Projektstrukturplan*') and finally the '*Netzplantechnik*'. Part four provides the framework for organizing a project, regarding leadership (PL) and teams. Finally, part five gives a definition of project phases and how to report completed projects.

The companion standard `ISO 10006` has been published in January 2004 additionally as DIN 'Norm'. It's scope is to provide a guideline for quality management in projects, broken down in eight chapters and comparable to the standard `DIN ISO 9001`.

2.4.4 SCRUM

SCRUM [57] (the word is not an acronym but rather originates from football), is a relative new software development framework considered as part of '*Agile Project Management*' [2] with a lot of inertia. Though is not an official standard, it can be understood as a particular 'school' or method. Unlike the classical or legacy PM method, the *project's subject* (we call that the '*product*') is not fixed at the beginning of the project, but rather the product's qualifications and attributes are 'formed' during development. In short, the product is determined by the project and not reversely. In classical PM the project plans are altered as much as it is required to meet the initial product design.

APM

2.4.5 SEI

The *Software Engineering Institute* **SEI** at the *Carnegie Mellon University* [22] is one of the most experienced organizations regarding process management and quality of software, and publishing their results since 1986. Not only, that they introduced the famous **SEI** *process maturity levels*, but rather, today they promote the *Capability Maturity Model Integration*

CMMI

(**CMMI**), which is another cornerstone in software engineering and stands as synonym for continuous process improvement.

2.4.6 ISO/IEC 12207

This standard, published in 1995, introduces the idea of a "*Software Life Cycle Process*", in particular suited for tailored software (unlike standard software). Here, the terminologies primary processes

- ordering,
- delivery,
- development,
- operation, and maintenance

the complementary processes

- documentation,
- configuration and
- quality management, verification, validation, and audit

and finally the organizational processes

- management and
- infrastructure

is introduced.

2.4.7 ISO/IEC 15504 (Spice)

PRM & PAM

The *Software Process Improvement and Capability Determination* **Spice** model [58] has been poured into the standard `ISO/IEC 15504`. Spice does not only provide a *Process Reference Model* (*PRM*) but in addition also allows to assess they achieved quality by means of a *Process Assessment Model* (**PAM**) and thus is able to determine the maturity level of a process, comparable to SEI's definitions. In particular, the automotive industry uses Spice as quality framework.

Background, Exercises, Facts

- Task 1:
  Visit the web pages of the project management organizations:
  - http://scrum-master.de
  - http://www.apm.org.uk/
  - http://www.pmi.org
  - http://www.pmi-muc.de/
  - http://www.prince2.com
  - http://www.ogc.gov.uk
  - http://www.sei.cmu.edu/cmmi/
  - http://www.scrumalliance.org/
  - http://www.isqi.org/
  - http://www.dgq.de/
  - http://www.gpm-ipma.de/
  - http://www.12manage.com/
- Task 2:
  Order that list according to the 'frameworks' (standards) discussed above. Complete the missing ones while 'googling' for them.
- Task 3:
  On some of those web pages, a short summary or method diagram can be found. Download and print these for later reference.
- Task 4:
  How to obtain the respective DIN or ISO standards?
- Task 5:
  Make a 'pyramid' drawing of the hierarchical breakdown of a project organization. Consider upper management, the PL, the team leaders, and the team.
- Task 6:
  Identify and attach the stakeholders to that picture.
- Task 7:
  Consider projects build up from several sub-projects. How shall this organizational scheme be extended to your opinion ?
- Task 8:
  How to characterize the rôle of a PM ? Is it more operational project management or more strategical ? Assign rôles to the project organizational levels.

Part II

Pre-Conditions for Project Management

Though *Project Management* is a cultural achievement, it requires particular circumstances to unroll it's effectiveness. These circumstances are usually outside the sphere of influence of the particular project itself, but have an important impact on it's success or failure.

Chapter 3

External and Internal Pre-Conditions

3.1 Business Plan

Projects, and in particular IT projects aim for a certain goal. This goal is typical a set of assumptions about the functional (technical) achievements of the project and the corresponding business advantages accompanying the realization of the project.

Depending on the project, this might be a direct and measurable economical benefit, or yield an indirect and perhaps a marketing advantage with respect to our competitors, or at least increasing the reputation against our customers, or supplier chain.

Today, it is a characteristic of an IT project, that the (technical) goals are pre-determined externally and do not depend on our behalf. Rather, we have to head for 'progress' because either competition forces us to renew our products constantly, or perhaps the public opinion or even standardization's and requirements for conformity pushes us to re-define our product, thus new releases come out frequently.

Good examples for these requirements are the ever changing versions of Microsoft's *Word*; requiring **ODF** capabilities and actually providing **OOXML** instead the open standard. Other samples are Adobe's *InDesign* with now 'Web 2.0' capabilities and or course the constant fight of all email software against malware and spam.

Thus, unlike a typically project, IT projects are driven from outside (except for a few). This makes management 'fed' about IT projects: "Why do we have to spend again .... money for this development; the old solution is working well". In turn, IT projects can be characterized to be

- market driven (by competitors, standardization requirements, quality standards)
- cost driven (reduce production/operating/maintenance costs)

- technical driven (introduce innovation and/or substantial improvements).

Considering the volume of IT projects under the current market situation, the economy, the ratio between these dependencies is about 6:3:1 to be optimistic. Or course, setting up IT projects may be driven by all three aspects. One example to mention is Apple's move from IBM's *PowerPC* platform to Intel *CPU*s.

3.1.1 Technical Evaluation and Market Analysis

The start of a successful IT project is a good understanding of the reasons why it should be carried out. The dependencies can be picked up from figure [3.1].

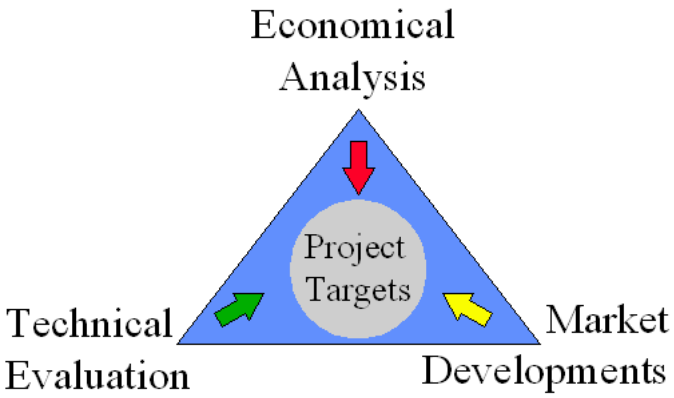

Figure 3.1: Dependency triangle for project targets

Business Plan

Upper management has to evaluate what fuels the forthcoming IT project and has to provide an impact analysis. A *Business Plan* has to be carried out, which incorporates the current decisions and provides a guideline for the Project leader and the Project team during every phase of the project. As indicated, while the project is running the goals and the dependencies to and how to finish the project may change, since neither upper management nor the Project leader has the definitive Crystal Ball to know the future.

On the other hand, a lot of IT projects focus on solutions for which the original assumption has already been superseded. In case of substantial deviations between the original targeted goals and those actually realized in the current project, a gap between the real project and the believed project opens up, which has important (negative) consequences for the project and perhaps makes the original *Business plan* invalid or obsolete. Certainly, this has an important impact of the recognition of the Project Leader, the Project team, and the project itself by upper management.

According to a recent analysis from the Standish Group (see figure [3.2]) [18], a significant percentage of IT projects is challenged or even fail:

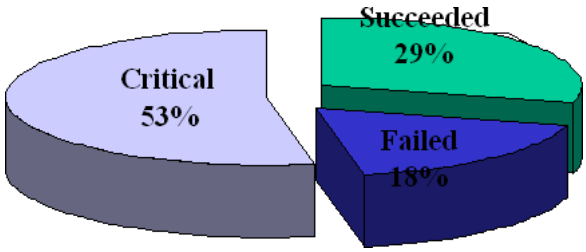

Figure 3.2: Chaos 2004 Survey Results [18]

The reason for this situation can be characterized due to

- external factors: competition, moving market situation, technical risks
- internal factors: limited resources, challenging time-line, underestimated budget

While the external factors hardly can be controlled, it is not uncommon for the market leaders to pet the market such, the acceptance of the new products is raised and thus the risk of a failure is reduced and of course the market shares are maximised. The *roll-outs* of new products from companies like Intel or Microsoft are often accompanied by such additional marketing mechanism.

3.1.2 Return-On-Investment Calculations

Let's assume an IT project has finished a certain software (product). From the company's perspective an IT project can be considered successfully, in case one of the following targets have been met:

- *Functional*: The product fulfils it's technical requirements.
- *Market share*: The products helps to improve the company's market position.
- *Economical*: The revenues realized by the product exceed the development costs (break-even point).

ROI

Any delays in the delivery of the product has significant impacts on all those targets. While functional and marketing issues can be considered 'soft' targets and the arising consequences are difficult to estimate, the economical dependencies can be estimated by means of a *Return-on-Investment* (**ROI**) calculation, as shown in figure [3.3].

Typical examples for such a calculation are the Roll-Out of Microsoft's *Windows 98*. Here it was clear from the beginning, that this update to Windows 95 (due to the difficult and almost complete change of the peripheral driver model [VMWD]) did not meet the required technical maturity, due to marketing constraints, the product had to be delivered in time.

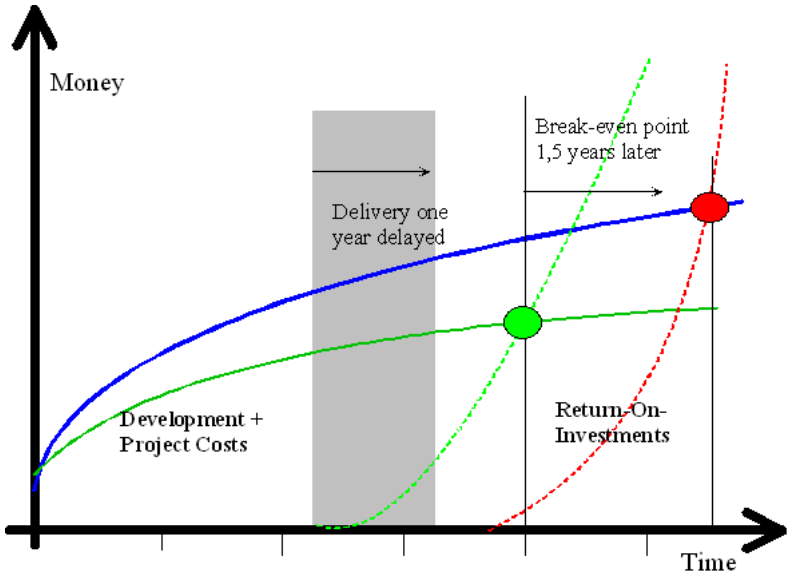

Figure 3.3: Return-on-Investment calculation [24]

3.1.3 The Mission Statement

Once the projects targets are defined and upper management has finally committed to provide the required resources (during the time, the project spans), the final 'Go' for the project has to be announced with the project's Mission statement.

The Mission statement has to fulfil the following criterion's:

- *Formal*: The wording has to be clear and concise; usual phrases have to be avoided (i.e. 'most important', 'lighthouse project'), the targets of the projects have to be named explicitly.
- *Content*: The project's targets have to be reachable; they have to be confined, thus do not depend on external circumstances.

The *Mission statement* should act as guideline for all project team members and of course the **PL** . Upper management has to convince everybody that they stand undoubtedly behind the *Mission statement* and that the success or failure of the project is considered their own success or failure.

3.2 Project Management in Conflict

In general, any (IT) project has to deal with limited resources and competition among other projects. It is one of the initial task of project management to identify not only the risks but in addition the potential shortages and conflicts for his/her dedicated project. Conflicts arise in case the resources don't meet the project's requirement, as detailed in figure [3.4].

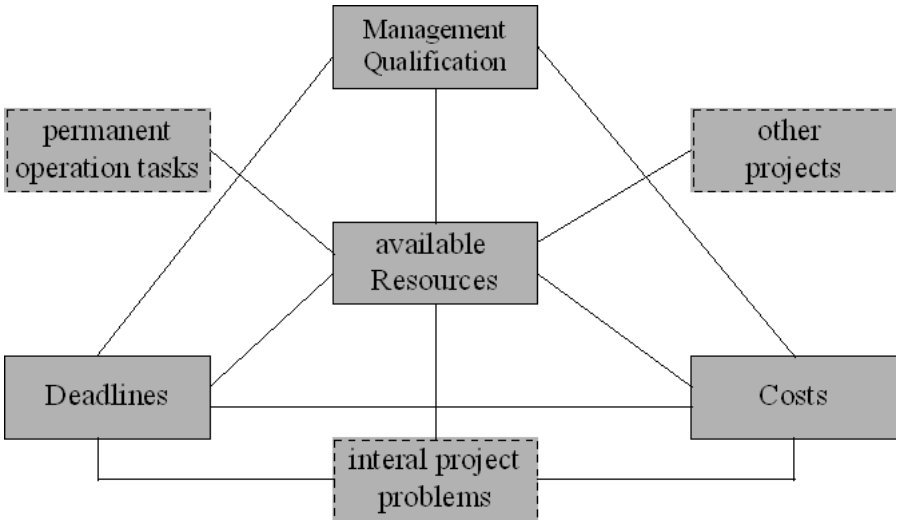

Figure 3.4: Sources for conflicts in projects [24]

While practically conflicts in particular among project (and staff) team members can not be avoided in the running project; conflicts with upper management are often more severe for any **PL** . Possible reasons may be:

- *Project targets* and their *deadline* are not clearly defined or deviations are not reported, thus hit upper management unprepared.
- *Allocation of resources* is not possible due to budget restrictions and/or the budget has already been used up.

Project managers should also consider during the definition phase of the project and before any commitments against upper management are carried out, what are the potential penalties in case the project does not succeed. In case no such penalties exist, the engagement for the project of in particular regular staff members may be reduced ('business as usual').

3.2.1 Balancing Budget, Deadline, and Quality

We have already discussed the impact of the Deadline with respect to the *ROI* (figure [3.3]). However, in IT project management it is common sense not only provide the product in time, but also to achieve a customer-acceptable quality. This includes not only to meet the functional requirements (as probably beaconed by marking already), but in addition to ship a product which is reasonable mature and considerable bug-free. IT project management is always under pressure to achieve quality under the current project conditions. This is know as the Magical Triangle of Project Management [3.5]:

Magical Triangle

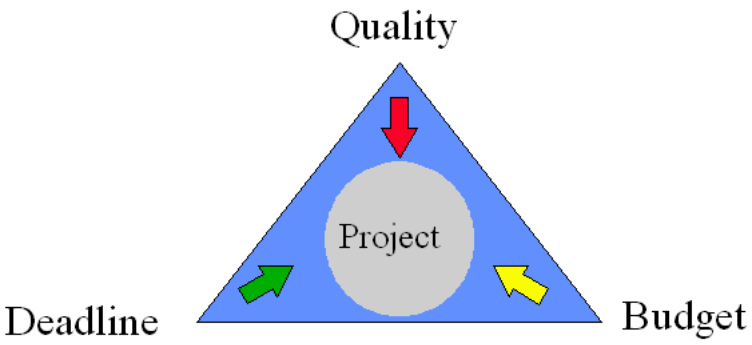

Figure 3.5: Magical Triangle of Project Management [24]

3.2.2 Effectiveness versus Efficiency

In order to achieve the defined goals, project has to be effective. However, the limitation of resources requires in addition to organize the project efficient. After a project step or phase has been finished, it is task of the *Controlling* to evaluate both, Effectiveness and Efficiency. On the other hand, there are no 'golden rules' how to achieve efficiency initially, except for a very few:

- Avoid complex project structures and project plans; complexity is often aligned with imponderability.
- Allocate resources when it is required and don't delay them; delay is typically a synonym for less efficiency.
- Identify problems and solve them as soon as possible; otherwise problems will turn into risks.

Figure [3.6] shows a breakdown of potential factors impacting efficient project management.

3.3 Project Conditions

Today's IT projects don't happen in a virgin environment; rather they are set up by companies which have already finished several other projects. Thus, the foreseen project leader has to make himself (or herself) familiar with the existing conditions the forthcoming project has to be carried out under [3.6].

We have to discuss these issues regarding

- the existing *technical framework*,
- the o*organizational circumstances*, and most important -
- the a*administrative competencies* and *reporting structures*.

Figure 3.6: Success factors for efficient project management [46]

3.3.1 Existing Technical Framework

While studying offers for IT project managers in the big newspapers often a particular development framework (like Visual Studio or Eclipse) is mentioned, for which the demanded project leader should have special knowledge. To be honest, either here a *sub project leader* (**SPL**) is required, or the *human research* (**HR**) department has no good understanding what the tasks of project managers are.

Certainly, the knowledge of the technical software development methods is beneficial and a deep understanding here allows to judge the potential risks and perhaps quality of the code produces by the developers.

Development frameworks

Integrated Development Environments (**IDEs**) are typical for certain computer languages. Today's, those framework allows to collaborate; thus developer share a common repository and the access may be client/server or web-based. Most common today, in particular for the programming language *Java* a lot of development frameworks exist which allow an efficient usage of the object-oriented features of this language.

The development frameworks may additionally include *software modelling* techniques which in today's understanding is **UML** (*Unified Modelling Language*). Those frameworks have to stay ahead the current state and version of the software modelling language. From here class and object hierarchies can be retrieved, which makes coding effective and efficient; though not guaranteeing that the code is efficient as well.

Frameworks are required as well in order external resources have to be includes. This is the case if PL/SQL code has to be developed, which is only

possible accessing directly the database instance.

Source Control Management

While occasionally IDEs include a (favorite) source control management (SCM) system, in general this is set up distinct. Typical choices are the client/server based systems Subversion or *CrossVC* to mention the public domain tools in the first place and IBM's (Rational's) *ClearCase*. Most of the systems provide a commonly shared repository to be accessed by UNIX and Windows clients, which is not uncommon for e.g. Java software projects. Of course these systems differ significantly the way the code is represented to the developer and how to derive a 'release' out of the sources which is called the '*build*' process.

Document Management

Though, it is not uncommon to store development documents in the *Source Control system* as well, more practical and more efficient is to use a dedicated system which allows to organize and to review the documents having one central repository. While developers have access to and work with their **IDE**, architects, quality managers, and stakeholders included for revisiting the documents require a separate document management system automatically changing revision numbers, indexing the document, and providing a history of changes. However, in practice often documents are send by mail, and the last version resides in a private folder without changed the document's revision.

In addition, a clear structure and reliable filing to the documents is required, not to search for documents to long. Thus the document filing should be realized in terms of **URLs** (*Universal Resource Locators*) which is commonly called hyperlinking. A good system achieves this not only per document but rather per section, in case the document templates are structured adequately and obey a defined hierarchy.

Quality Management

Quality Managers can be set up in the current project as sub project leaders or they may be organized separately. In this case, they report directly to upper management and don't need to follow the advises of the PL. Typically the quality of software is tracked by a special tool, which is essentially a database application with a (Web) front-end to enter and to follow bug reports. In software development terminology, deviations of the tested software with respect to the specification is called a '*defect*'. Such systems (ie. HP *Quality Center*) not only allow insert defects, and to define a 'lifeline' for it in terms of the attributes priority and status, but also to combine defects with the same source.

While the single defect is important for the individual developer, the regular obtained quality report, showing the distribution of defect priorities and correlating them with the sources tells much about the current state of achieved quality.

In order to obtain a complete picture, not only the individual reports are of interest, but rather how the development of defects took place, in terms what has been tested and how intensive the tests were done. A complete picture can only be obtained, if in addition the *Design Documents* are

considered and the complete *Use and Test Cases* are filed (see figure [3.7]).

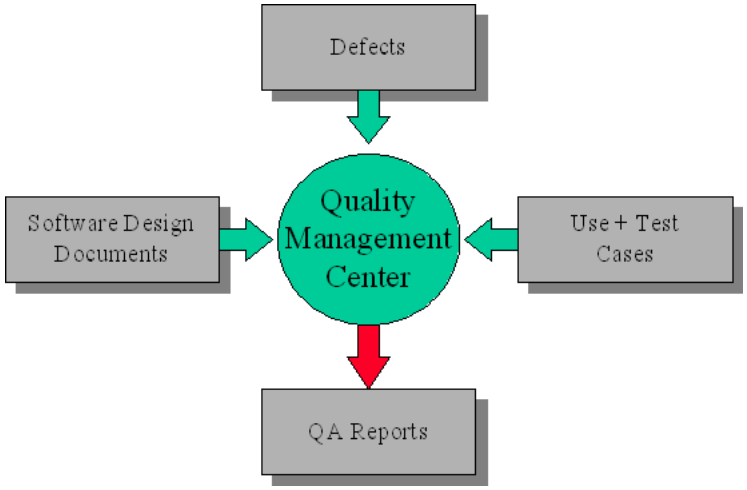

Figure 3.7: The role of the Quality Management Centre as Hub

3.3.2 Existing Project Organization

Entering an existing project team as PL, you first may want to see and judge how this team performed in the past. The best way is to check the way the Project Office works, how do people report their activities in terms of Time Sheets, and perhaps which projects were already completed and in what state.

Project Office

In order to achieve continuity a well organized *Project Office* (**PO**) is indispensable. Filing of documents, ordering of trips and hotel rooms, booking meeting rooms, account management, managing the team's agenda, preparing meeting documents, and many other activities are on the list of support activities.

The PO may act in addition as *Centre of (informal) contact*. As project manager make yourself familiar with the organization of the PO, streamline document filing and co-ordinate important agenda items with the responsible people here. Common and repetitious tasks should be delegated to the PO, which in turn has to understand it's task as Project Assistance and not just administration.

Time Sheets

While the project is ongoing, the project members have to report their activities on time sheet lists, in order to monitor the time required to complete a specific task. Of course, keeping the Time Sheets to-date takes attention and time and depends on the granularity of the reports.

Together with the SPLs, the organization and the items to be subject for the time sheet has to be defined and communicated. Limits on the particular items (in terms of project hours) have to be included and the project's progress has to be correlated with the reported hours.

The time sheets are subject of budget control and thus have to be reported to upper management. In addition, a good on-line reporting system, which has hooks to the IDE and perhaps the QA system, relieves the team members from stupid recording of the actives and in addition improves the level of correctness. The scope here is, not to control the team members bur rather to identify budget and time shortages.

Project Heritage: Styles, successes and failures

It is certainly a good idea to gather information about successful and less successful projects realized by the company or under the current project organization. The bottom line here is the quality of the information. A good project organization would provide a project completeness report; a badly organized project probably will leave the remnants of its work (directories and files on the file servers, code in the Source Control system) unchanged and visible for everybody.

In addition it is worthwhile to interview the former project members and perhaps managers for '*lessons learned*'. For any company and organization it is important to steadily improve the quality of project management. This can be done by internal and/or external course, allowing project members to achieve certificates, and file important documents and templates as guideline for future projects.

3.3.3 PLs Competence's

Split responsibilities

One of the most important tasks of (new) PL is to discuss and receive the necessary competence's from upper management. Bigger projects have at least an *administrative Project Manager* being responsible for *risk management* and in general controlling, and a *technical Project Manager* in charge of technical co-ordination and quality management.

Typically, several Sub Project Leaders (SPLs) are foreseen, dedicated to co-ordinate fixed tasks and reporting PLs.

In addition, a *Chief or Business Architect* can be established, who is responsible to co-ordinate development with *Demand Management* or *Marketing* and perhaps acts as co-ordinator with respect to customers or partners, to whom interfaces are designed.

Reporting Chains

Apart from 'managing' the project, the PL has to report the state of the project to upper management. Two types of reports shall be considered:

1. *Standard reports*, defining in a defined and concise way the current status of the project, w.r.t. to particular sub-projects or fields.

2. *Exception reports*, which need to be raised in cases the project runs suddenly out of plan, a particular risk has been identified, or the team (project) suffers due to a substantial crisis.

Reporting chains and acceleration schemes have to be clearly defined, in order not to deadlock the project.

It is the bare responsibility of the PL to react 'well-behaved' during a crisis: regarding the project team and in particular concerning upper management.

The current financial crises even of large international banks show clearly their incompetence setting up and utilizing corresponding exceptional reporting channels.

Budget Planing, Recruiting, Ordering

While the budget for the project has to be allocated by upper management, the (efficient) spending of the money is one of the responsibilities of the PL. Responsibility might not coincide with competence.

Considering the financial foundation of the project, it is important to understand the following distinctive sources of the project budget:

- **CAPEX** (*Capital expenditure*) [48]:
  For the budget a certain fixed amount of money is allocated to be spent until the project is finished. In particular, external consulter's (like developers) are hired under the condition of a limited budget or time frame.
- **OPEX** (*Operational expenditure*) [54]:
  This money is taken out of the budget for general operations. Internal project team members may be funded by OPEX means.

It is not uncommon for a budget to run on both CAPEX and OPEX means; however for internal reports (time sheets), this differentiation might not be required.

In case the projects does not exceed the allocated budget, certainly it is up the PL to decide where to use the money for. s outlined this situation characterizes perhaps 2/3 of the project lifetime. Overspending is common for most IT projects. Whether upper management is willing to put more money into the project depends definitively on a convincing PL. It is certainly helpful, if the project management made up their homework and could provide that the project until now is under (budget) control, the financial risk of not spending the expected amount of money to (almost) finalized project is higher rather than allocating N more Dollars or Euro to the ongoing project.

Project Controlling

As discussed, the main function (according to upper management) of the PL is to control the ongoing project and to focus it in the interests of the company. In effect, controlling is a rational method to estimate risks; thus controlling and risk management's are twins. Of course there exist a thin line between real risks and reported risks. The example of the current bank and financial crises shows, that real and reported risks might not coincide:

- In order to estimate real risks one needs measures.
- In order to convince upper management one needs to present established methods.

In the situation of a PL, an underestimated risk may break the whole project, while an overestimated risk may block it. Here, a good relationship to upper management is one key to survive and eventually master the crises.

3.4 Project Initialization Summary

This chapter approaches to summarize the necessary tasks required to *Initiate a Project*. A *mission statement* together with a business plan are considered to be the most prominent documents describing the project, it's funding, and the project's subject: The product. The picture is incomplete without taken into account the current project management and project operational circumstances and conditions. Since the Project Manager (PM) itself is one of the most vital parts of the project, his (her) responsibilities and rôles needs in particular a clear definition.

Figure 3.8: Supertramp: Crisis? What Crisis?

Background, Exercises, Facts

- Task 1:
  Identify the essential steps to set up a project (we call that later the 'Project Initialization' phase).
- Task 2:
  What is the 'ranking' of those steps?
- Task 3:
  Who is determining the project running conditions? The PM or the organization? Discuss potential conflicts and the responsibility of the PM !
- Task 4:
  The original financial funding of the project very often is insufficient to complete the project. 'Google' for the name of the project management 'discipline' which is assigned to !
- Task 5:
  'Google' for the terms CAPEX and OPEX. What is the main difference between both financial sources regarding their tax relevance?

Part III

Team Management

> Do all the hard parts of management: Setting up teams, coaching, adjusting people's assignments, deciding which risks to take and which one's to avoid, resolving conflicts, hiring, and motivating people. This is the guts of what real managers do.
>
> *The Deadline* – Tom DeMarco

Chapter 4

The Project Manager and his Team

From the previous chapter we have seen that Project Leaders (PLs) or Project Managers may have different tasks, in particular whether they are assigned more administrative or more technical rôles. However, apart from their specific tasks, there is one main skill they should have:

Leading people in a team.

Team management is a key for effectiveness and especially efficiency. If the team is irritated or spoiled because of bad management, only very little output can be expected. Rather, the individual members of the team follow their own interests or perhaps fight against each other, or (individually or commonly) against project management. Thus, to streamline the individual interests with the project, and to put the team in a position to actively anticipate (positively) the commonly achieved results is a main factor for motivation.

Though it is possible to stimulate a team using 'psychological means' and to convince team members in particular during special workshops/events, in the long run only a qualified team management, balancing the project requirements with the current state of the team members will yield sustained results. To turn it the other way around:

> *Once project management requires and recruits highly skilled people, turning the tables, exactly those people will demand a project management which acts professional and takes their requirements seriously.*

Those team members are Stakeholders by definition and will impact the results of the project significantly. Therefore, we will discuss the following general tasks for project management:

- Setting up teams

- Organizing the team's work
- Reacting in case of conflicts
- Supervising the team's work

4.1 Setting Up Teams

4.1.1 Organization of Project Management

After management has made it's mind to start the project,

- a business plan has been established and the first ideas about
- a mission statement have been put in paper, and even before
- a concrete project plan has been worked out, the first impression

of the complexity and the basic requirements of the forthcoming project can be guessed.

After having answers for the basic questions:

- What is the deadline of the project ?
- What budget is required to successfully run the project ?

this should be the starting point to define the least (minimal) management structure of the project; and thus how to shoulder the tasks, which organisation the project should have, in particular regarding

- physically office spaces and required material (PCs, Notebooks, Servers, Software, etc.),
- logistics project office, and most important
- administrative reporting chains.

These decisions should be mutually carried out with the project managers; if already designated. Of course, during the lifetime of the project these conditions are subject for change. In particular, the PL should be free (and entitled) to appoint certain technical tasks to SPLs. Reversely, changes in the other aspects during the running project will have an impact on at least the deadline, though it might be necessary to improve the project conditions significantly on demand, in order to meet the required objectives.

4.1.2 The Project Manager

Generally, it is not easy to try to define what somebody makes a 'good' project manager; and how to convince management that you are the one to choose. What is required for project management ?

- The best qualification is practice; once your résumé includes a reference regarding the (successful) leadership in a comparable project, you are fine. For newcomers, certificates in the project management method required/requested for the current job (PMBok/PRINCE2) is a must. — Qualification
- Since IT projects are technical driven, expertise in the currently employed techniques is required to master the different tasks. — Skills/Know-how
- Project Management is in the first place risk management. Risks can be mastered successfully only with a certain of amount of blood, sweat, and perhaps tears. Only if you are willing to pass those phases, the project will benefit and eventually complete successfully. — Dedication
- Upper management has appointed you to realize the expectations of the company; which might not in every phase coincide with the interest of the project. Loyalty has different sinks: Management, team, project tasks. Be aware 'sitting between chairs'. — Loyalty

Be prepared to fulfil several rôles as PL:

- You are the *manager* of your people in charge, the better you do regarding your task, the better they will perform. — Leader
- A project manager has to have not particular more insight into specific project items, but should have a broader and more coherent view. Thus, your decisions are expected from the team to reflect this; thus anybody finally accepts those. — Guru
- Your are the *agent* of the company's management fulfilling the tasks of — Agent
- a team (and reporting as leading staff member), run an *organisation* and realizing the projects objectives (defined by sponsors) — Organiser
- It is important that you don't forget about *yourself*. Sincerity is a key for authority. The complementation of *leading* and *organisation* is tentatively shown in figure [4.1]. Of course, while applying for the job as project manager, make up your mind whether *Yourself* fits with *Dedication* and *Loyalty*. — Yourself

On the other side, your interviewer will decide whether your referenced *Qualification* and your *Skills/Know-How* meet the projects requirements. Apart from specific qualifications/skills, for *System Engineering IT projects*, it is expected that the PL incorporates the following management concept (figure [4.2]):

4.1.3 Leadership models

Any IT project management leadership models have to concern that management takes place in an complex environment, which is determined by (see figure [4.3])

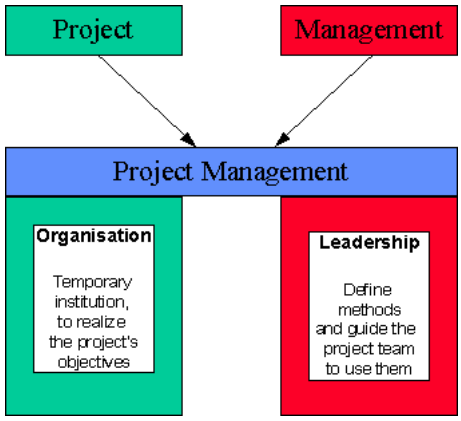

Figure 4.1: Complementation of Leadership and Organization in Project Management [24]

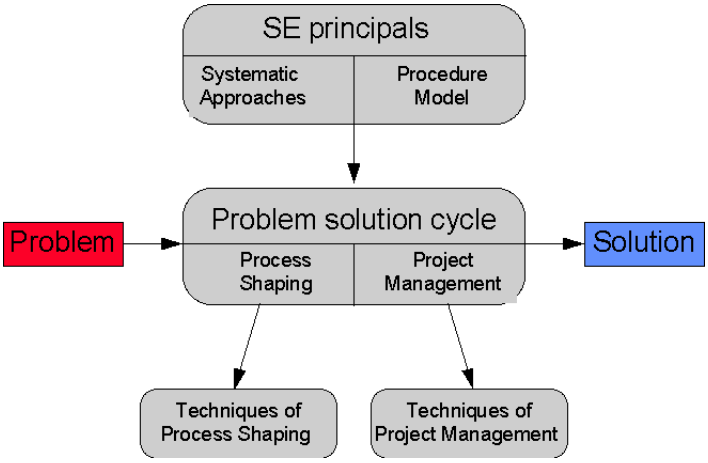

Figure 4.2: Project Management concept according to System Engineering (SE) principals and guidelines [24]

- the PM organisation (already existing to be build)
- the chosen PM methods (see next section),
- the supporting software tools for PM (discussed in the forthcoming chapters), and
- the (human) leadership capability of the PL.

According to our current understanding, the project management leadership model has to obey a systemic approach, as laid out in figure [4.4]. Effectively, this is a 'people business' which takes into account

- the different qualifications,
- the cultural impacts (of multi-cultural teams), and
- the individual (human) interaction behaviours

of the team members.

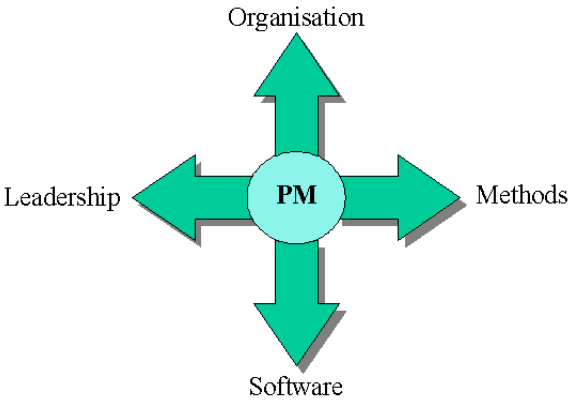

Figure 4.3: Determinants of Project Management [10]

4.1.4 Declaring PM methods

We have already realized, that the choice of the project management methods is very often pre-determined by the heritage of the company, or depends on external circumstances and the nature of the project, or perhaps is already defined by the sponsor of the project (`DIN 12207`, `DIN 10006`, `ISO 15505`).

Of course, those models have to be adjusted with the current project and a particular procedure model has to be carried out. This has to be explained to management and sponsors and finally has to be agreed by them. In addition, it is not only necessary to inform the project members about the chosen procedure model, but in addition to let them participate and to further develop those methods. This participation could be done by means of a kick-off meeting or workshops, which should aim to actually apply the methods so sub-projects and tasks already appointed to the group.

Kick-off meeting

4.1.5 Building a team

The project team will be eventually completed during the first phase of the project and is perhaps subject of substantial changes during the lifetime of the projects. The interaction of the team members during the project impacts progress and outcome of the different tasks. Project leaders should be aware that in particular a newly build team acts differently according to the level of knowledge between each other [24]:

1. Formation phase:
   The group members try to check each other. Since the current environment is still unknown, often the PL is requested.

2. Conflict phase:
   Self-organized sub-groups are established; one knows about the qualifications/reactions of each other. Disputes among groups raise and

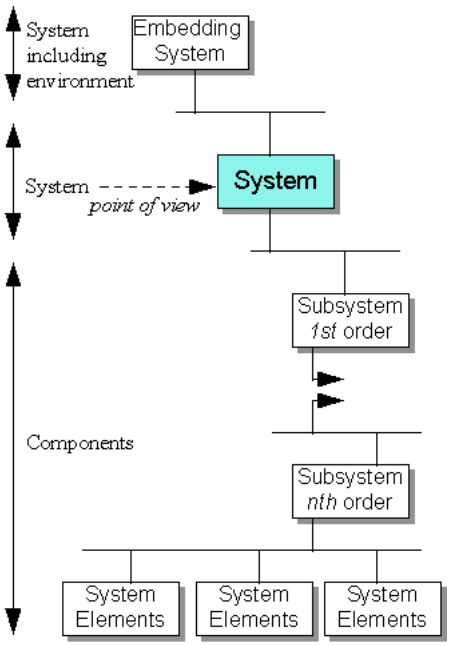

Figure 4.4: A systemic approach for complex systems [10]

occasionally, even management is questioned.

3. Normation phase:
   Disputes are settled, friction is reduced, and mutual acceptance is established.

4. Working phase:
   The power of the team can now focus on tasks. The rôle of the individual team members is fixed and working. Perhaps some members are heading for more challenging tasks.

A qualified team, able to master even complex tasks and reacting coherently and rational in difficult situations, can be characterized by the attributes (see figure [4.5]):

- A small group of individuals with complementing skills,
- motivated and driven by a common task,
- willing to achieve the same results,
- while co-operating with engagements, and
- mutual responsibilities.

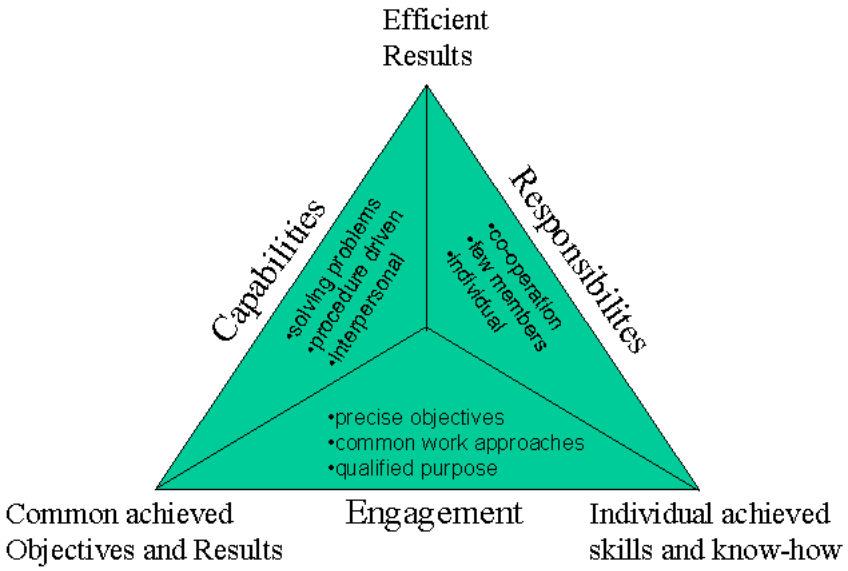

Figure 4.5: Team building foundations [24]

4.1.6 Delegation

Delegation is a key feature of leadership. During the project, the PL has to delegate some of his/her activities and responsibilities to other project members. Two types of delegations are common:

- Permanent delegations
  Here, the PL entitles Sub Project Leader (SPL) to fulfil defined (sub-)tasks, mainly as result of a *Work Breakdown Structure* (**WBS**). Also, project management can be realized in a team, perhaps in conjunction with project assistance or with a junior project manager.
- Temporary delegations
  Are used typically for restricted tasks which can be finished in a certain time frame. In both cases, the PL as to clearly announce the delegations and to officially entitle the project member among the team.

4.1.7 Skills and People CMM levels

In order to asset the skills of project teams, the **SEI** at *Carnegie Mellon* has adopted their *Capability Maturity Model* to people (*People CMM Version 2.0*). Scope is not only analyse the current level of maturity (qualification) but rather to allow management CMMI

- to set up programs
- to improve the competence of the individuals,
- to make team-work more effective,

- to motivate team members and to raise their efficiency,
- to organize the groups in order to prepare them for the forthcoming tasks.

According to this model, organizations can be assessed in the following way (figure [4.6]):

1. Initial Level
   Undefined process flow, unclear competence's and responsibilities; ritual procedures; team not involved.

2. Managed Level
   Members work to hard, personnel objectives are unclear, missing knowledge about solution concepts, restricted communication, bad moral habits.

3. Defined Level
   Missing streamlining between groups, thus little synergy. PL is not capable to identify skills of team members and to efficiently incorporate those into the project

4. Predictable Level
   Project/Organization employs skills of all members. PL can predict from the teams involvement the results of the project in terms of quality and deadline.

5. Optimizing Level
   Management tries to improve continuously the skills of groups and team members by analysing the level 4 achievements.

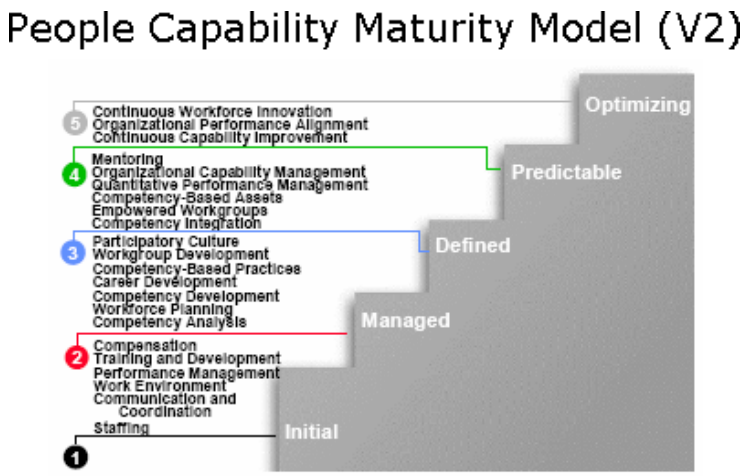

Figure 4.6: SEI's People Capability Maturity Model [1]

4.2 Organizing Teams

4.2.1 Project Office

The Project Office (PO) has already been introduced as

- Centre of administrative tasks,
- location of work-flow sheets and perhaps other document, and overall as
- communication Centre for the project.

Practical experience shows, that the involvement in the project is related to the distance of a project member's office to it. However, if team members are located to close, their efficiency is getting reduced by the busy PO (or other circumstances). In addition, the PO should display the master plans regarding the projects or sub-projects progress and perhaps a list of open tasks, a *List of Open Points* (**LOP**) respectively.

LOP

4.2.2 RACI Matrix

Common in projects is to assign certain tasks to persons. Complex tasks however, require the involvement of many people, team members (or externals) respectively. One approach to make the assignment transparent is the use of the so-called RACI matrix (figure [4.7]).

RACI is the acronym for '*Responsible*' '*Accountable*' '*Consulted*' '*Informed*':

RACI

- Responsible for execution.
  She/He is the *Master* or *Process Owner*: Person initially required for the execution of the task, everybody else has to report to the *Process Owner*.

  Responsible
- Accountable for sponsoring.
  Person providing the budget for the task and required for it's funding, aka the *Budget Owner*.

  Accountable
- Consulted for additional know-how.
  Person, which directly impacted by the task/project. In case the task/project depends on him/her, he/she acts as *Supervisor*.

  Consulted
- Informed for the current state of the project.
  Persons, which need to know the results of the project/tasks, maybe because they are *Stakeholders*.

  Informed

In the RACI approach, persons or organizations are assigned a particular rôle. Apart from the basic RACI approach, there are several variations in public use:

	Management	Project Office	Architects	Team A	QA	
Definition	A	I	R	I	C	
Introduction	I	R	C	I		
Step 1	I	I	C	R	I	
Step 2	I	I	C	R	I	
Step 2a		I	I	R	I	

Figure 4.7: Typical set-up of a project's RACI matrix

- RACIS *Supportive*
  required for supporting the project/task with particular resources.
- VARISC *Verify* and *Sign Off*
  necessary for evaluation and verification of the task and responsible for the final go and roll-out.

4.2.3 Meetings

Every project should have in addition a qualified meeting room, being permanent available and to use easily without the requirement for a long-term scheduling and booking. Here, additional communication facilities (for eg. video conferencing) should be available. Though today's network capabilities makes it possible to use any PC for conferencing; a meeting where people are joined in a common room introduces significantly more 'group dynamic' since it is much harder for anybody to escape.

Apart from the meeting room, the acceptance, results, and efficiency of a meeting depends strongly on the following criterion's:

Schedule in time
- The meeting has to be scheduled in time , PO has to make sure, that the meeting is not conflicting with other major tasks.

Well prepared Agenda
- A well-prepared agenda has to be circulated which needs to be streamlined with the SPLs.

Assigned Meeting Leader
- A meeting leader has to be assigned (perhaps in a turn-around fashion) and to be made familiar with the agenda and it's objectives in a preparation meeting.

Defined Objectives
- For the main items, objectives have to be defined and the project team and/or particular members/groups have to be assigned to those.

Concise Minutes
- It is important that minutes include these objectives in a concise way and reports about the achieved rustles have to be part of the forthcoming meeting agenda (when they are due).

- Minutes have to prepared in a standard scheme , they have to be circulated and filed; perhaps with additional comments from the PL and the group members. Filing Minutes
- Meetings should be clearly organized and treated as important subtasks. Otherwise, they will become obsolete, increase the level of frustration, and steal of course project time. Well organised

4.2.4 Document Filing

A reliable document filing is necessary to increase the efficiency for the internal project communication allow sponsors and upper management an undigested control of the current project's state. We have already heard about the 'no-goes' in document filing. However, what is the correct approach ? Here we have to consider three important ingredients:

1. An extensible and hierarchical document location structure. Location
   Documents have to be filed according to subject.
2. Adequate document templates. Templates
   Allowing an uniform an easy movement in the document.
3. A qualified Document Management System (DMS). DMS

Providing automatic re-visioning of the document, building up indexes, and allowing different '*views*' (by subject, owner, version, topic etc.) of the documents under it's control. One approach for consistent document filing was developed by IBM is known as **WSDDM** *Worldwide Solution Design and Delivery Method* [4.8]. WSDDM

Meanwhile there exist several public-domain products for document management, eg. *Tortois* [43] which either use a Revision Control Systems (here: Subversion) in the background or directly a RDBMS (eg. MySQL).

Another approach is to use a *GroupWare* solution to allow this document handling. In the case of IBM, the natural choice (until know) is *Lotus Notes* [19]. Microsoft provides on the other side a tools called *Sharepoint* [27]. However, in addition public domain solutions (like *eGroupWare* [15]) provide at least parts of the required capabilities. Groupware

Certainly, storing important documents in the '*Cloud*' is not a useful option.

4.3 Running Teams

4.3.1 Conflicts

The sooner, the later, every project runs into a substantial crisis. We define a crises as substantial deviation from the project plan and objectives in terms

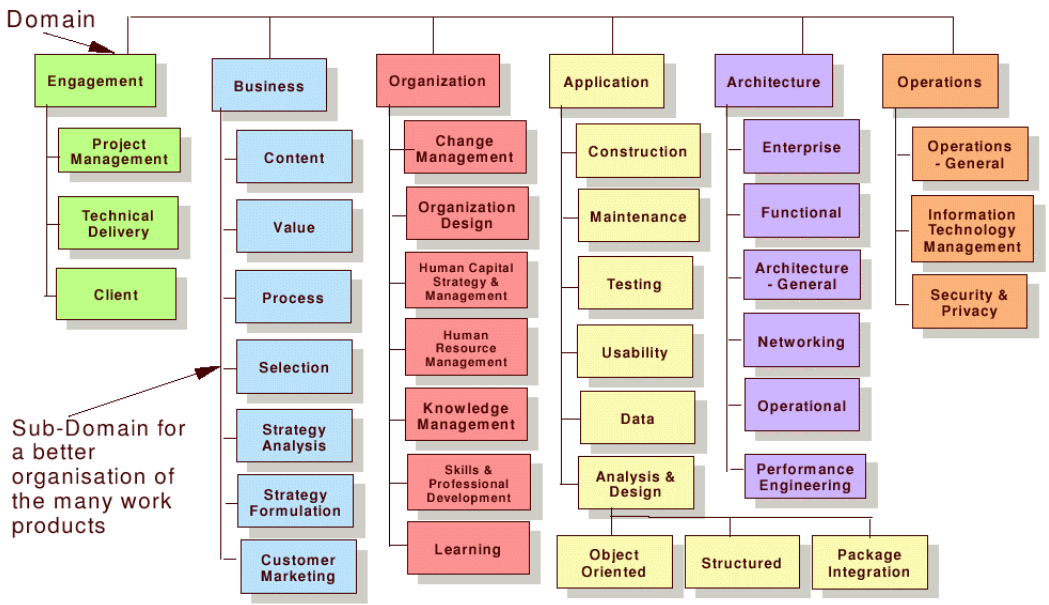

Figure 4.8: Structure of IBM's WSDDM model [42]

of deadline, or completeness, or quality with respect to the experienced or guessed current state.

A crises might originate from a variant of different sources; in particular internally and externally. One important reasons for a crises might be conflicts in the team, yielding a substantial reduction of productivity. Figure [4.9] provides a breakdown between the productivity of groups and their cohesiveness:

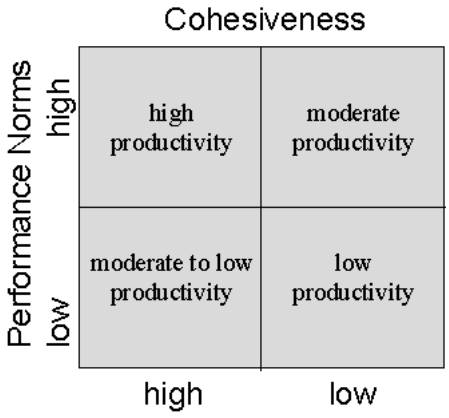

Figure 4.9: Correlation between group cohesiveness and productivity [11]

4.3.2 Workshops

Workshops are useful in case, the crisis originates from technical or organizational problems. In order to successfully complete the workshop, a script should be prepared defining steps and rôles. Depending on the foreseen results, the workshop should take place outside the 'work arena' and perhaps off working hours. Since this includes additional attention and efforts from the team members, it has to be prepared with management attention.

4.3.3 Mediation

It might be necessary to ask for support from outside the group, and perhaps involve a mediator. A *mediator* acts always exceptional and outside all reporting chains, thus the team members should feel free to tell their concerns without any consequences. Mediation should take place unlike meetings and workshops only with the groups or team members in conflict and happen always none-disclosure.

4.3.4 Coaching

In particular for junior project members it might be good idea to organize a 'god father' and to coach those members respectively. Unlike the mediator, the coach is as member of project team familiar with the demands and methods of the project. Depending on the work load the coach could be recruited from a different group.

4.4 Controlling

4.4.1 Confirmations

The sub project leaders (SPLs) have to report directly to the project leader. One of the tasks of the PL is to confirm their reports, to balance the results with the original or modified project plan and to finally accept the work in terms of intermittent mile stones. Typically, the project is broken down in phases, know as *Work Breakdown Structure* (**WBS**). We will later see, that *Gantt* charts are a typical mean to express phases and dependencies.

4.4.2 Auditing

One mean to improve quality is perform audits on demand or perhaps permanently. For IT (software) projects audits are complementary to QA (Quality Assurance) reports. While the latter focus on reported bugs (defects) according to the test or use cases, auditing actually takes care about the source code itself.

SOX compliance

Auditing is required in critical cases. One of those cases is to audit the security impacts of the code and to verify the accepted methods are consistently used. Another requirement may originate from SOX 404 compliance [56]. Again here, the software has to be written in spirit of the law, and it might be necessary to achieve compliance by means of a certificate due to an external audit.

4.4.3 Reporting to Management

The Project leader has to prepare (at least) two different standard reports:

Progress Report

- Progress Reports
  This report has to detail the current state of the project regarding compliance with plan in terms of the achieved objectives and whether and not the residual aims can be reached.

Business Report

- Business Report
  The PL has to declare and to explain it's expenditure and to correlate that with the expected and forecasted spending. Here, the PO can play a vital rôle in order to gather the required information:
  - Summing up the team member's time sheets [4.10] per sub-project and project phase.
  - Provide an overview about other spendings in terms of operational costs, third party royalties and external contractors.

  It is obvious to mention, that qualified (and verifiable) reports are import in cases a critical situation of the project is encountered and a

Exception Report

- Exception Report
  has to be provided, whose purpose is to raise upper management's attention about the project and enable mutual solutions for extraordinary situations.

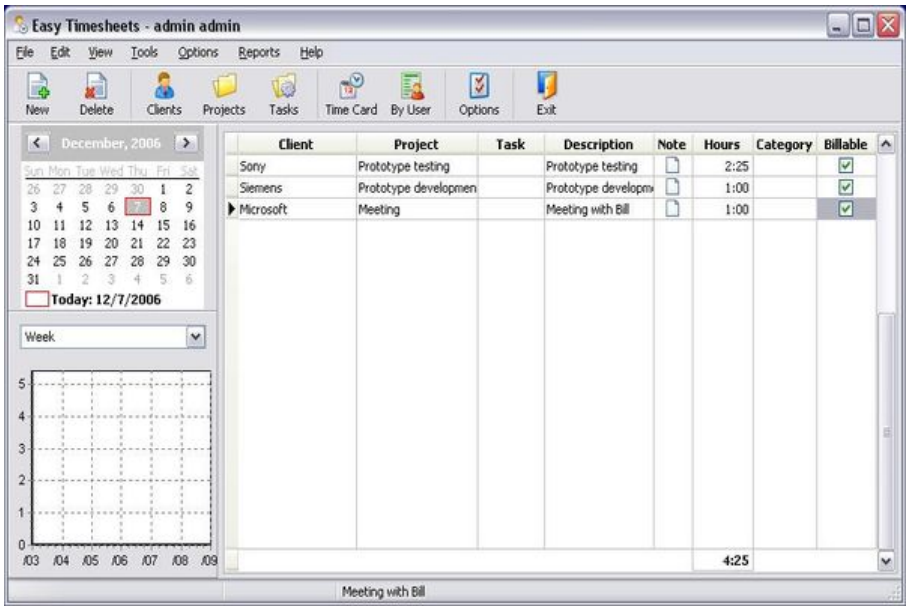

Figure 4.10: Reporting project activities in a Time Sheet [40]

Background, Exercises, Facts

Project Meeting Checklist:

1. Cause:
   What is the cause for the meeting?
2. Goals:
   What are the relevant goals to be achieved in that meeting?
3. Participants:
   Who needs to attend the meeting?
   Who shell lead the meeting?
   Who is writing the minutes?
   Who needs to receive the minutes apart from the participants?
4. Agenda:
   Whom to consult preparing the agenda?
   Who is responsible to distribute the agenda and invite the participants?
5. Meeting Invitation:
   Goals, Agenda, Location, Date/Time, Duration, Participants.
6. Material:
   Minutes of last Meeting, Proposals/Materials (printed), List of Open Points.
7. Facilities:
   Meeting room, Presentation equipment, Coffee/Drinks+Food, Accommodation.

Project Exception Report		
Project:	**Project number:**	**Date:**
Exception:	☐ Project goals are in danger ☐ Quality goals are in danger ☐ Project Deadline in danger ☐ Project budget exaggerated	
Exception cause and background/history:		
Proposed solution means:		**Consequences of proposed solution:**
Circumstances to consider:		
Decisions taken:		
Project sponsor:	**Project manager:**	**Date:**

Part IV

Project Planning and Scheduling

The initial *Project Idea* often takes places under some *special circumstances* outside the control of *Project Mangament*. However, the first steps – the subsequent *Project Planning* – are extremely important to identify the *feasibility* and the *scope* of the project.

Chapter 5

Scope and Tools of Project Planning

5.1 Start of a new Project – the Project Plan

You are assigned to be the new Project Leader. You've been introduce in the objectives of the new project, the time-frame, the budget you can spend. Management has shown you some of the most valuable team members, the office spaces and the IT systems currently in use for current projects (and in parenthesis you are expected to be knowledgeable in and to use too). At that point, you are probably knowledgeable about your duties how to realize the *Project Management Cycle* (figure [5.1) for the actual project.

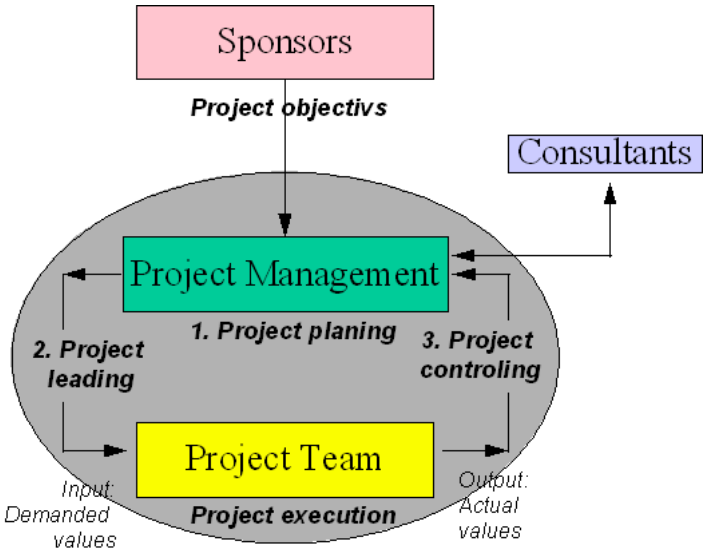

Figure 5.1: The Project Management Circle [46]

Depending on the assigned tasks and your experience, you may find your-

self between the following extremes:

- You feel a little bit like 'Alice in Wonderland' and you begin to realize the difficulty of your new job.
- You can initially guess that the means, sponsors and management have assigned to you, are fairly inadequate to finish the project as desired.

Probably it somewhere in-between and you sit together with those people you are believe to be Stakeholders for this project and discuss the amount of work and perhaps the organization of it.

As you know by experience, a project is nothing you can plan in detail and in advance, since it is driven by mostly inner forces. Thus, the first drafts) of the

- *project's tasks*,
- it's *Work Breakdown Structure* the it's *organization* in terms of Sub Project Leader and staffing

are undoubtedly vague and probably unrealistic.

There is still no good understanding, what efforts in term of time, labour force, and money needs to be spend for the identified legs. In addition, any risk assignment is still impossible. On the other side, a lot of people expecting already right now some answers from you about the project. While you need some time to digest your impressions, already now you are involved in organizational items, regarding Computer Accounts, required file spaces for project documentation, allowed absences of the project secretary, and others.

At that point, any project has a certain amount of entropy with lots of open bits and pieces regarding scope, organization, and time-frame. Thus, the first step as project manager is to reduce the entropy regarding the team members, and by the same token in your mind. In short, the first step is *to set-up a project plan* . Even this very initial phase requires internal and perhaps external expertise. Thus in combination with the foreseen planing step it is necessary to dedicate man power here, as laid out in figure [5.2].

Planning the Project Plan

Depending on the size of the project, this initial planing phase spans weeks or perhaps months and may involve much more man power as can be depict from the above figure. However, depending on the progress of the project, already some of the evaluations have already be carried out even before the final project manager has been assigned.

Thus, in that case the PL has to execute the task and participation is reduced to later adjustments. At the end of the initial step a *Project Plan* has been established.

This *initial Project Plan* (or *Masterplan*) will be accompanied by individual and more detailed phase plans. While the project walks through the

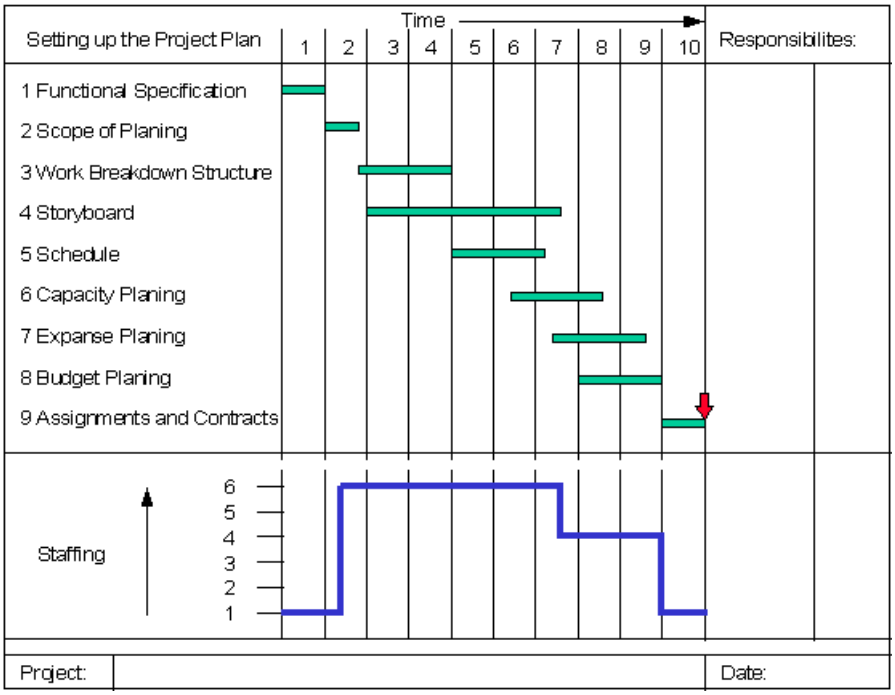

Figure 5.2: Setting up a project plan [24]

respective phases, the Project Plan is subject for subtle or more severe corrections. Thus, the Project Plan has to revised by Project Management and revisions have to be streamlined with the SPLs. The whole plan need confirmation by upper management and finally will be brought to attention to the project members. Figure [5.3] shows a typical evolution of a *Master Plan*.

5.2 Identifying Project Dependencies and Tasks

Any current IT project has to obey a lot of dependencies:

- Internal (given by the project itself):
- project definition (functional description, deadline, quality),
- realization (team. methods),
- budget constraints (costs, ROI)
- External (the resulting outcome of the project has to be placed in the real world):
- compatibility (interfaces),
- acceptance (customers), and or course
- competition (price, market share).

Without a good understanding of both aspects, a project runs on undefined risk which may impact the project significantly. Project management

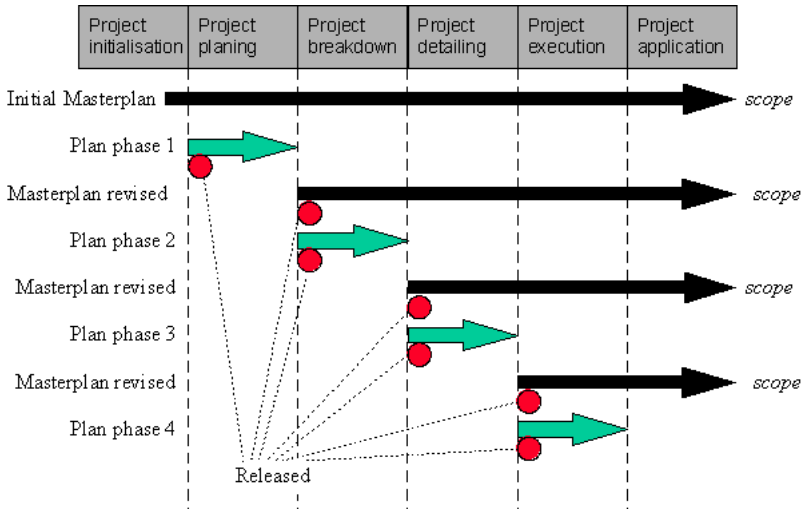

Figure 5.3: Evolution of the Project Plan during the project's execution [24]

is responsible to identify at least the internal dependencies, to express them explicitly against upper management and sponsors, and considering these in the project plan. Figure [5.4] provides in a bird's view the (mostly internal) dependencies to be considered.

5.3 The Work Breakdown Approach

Once the dependencies are identified, it is time to start working on the *breakdown of tasks* and to establish the *Work Breakdown Structure* (**WBS**) based on the *project's functional and/or technical description.*

The WBS is effectively a breakdown of the System Structure and allows to

- *determine* and *define* the particular functional parts of the system,
- *set-up* (confined) sub-projects according to the functional parts,
- *identify* dependencies and required interfaces between the parts, and perhaps
- allows to *assess* efforts and risks to the individual parts.

Typically, the WBS is a top-down approach starting from the general specifications and requirements to the necessary level of detail: The structural elements.

As with most plans, the WBS will evolve during the project's lifetime. In particular, in the beginning, not all functional parts can be clearly be identified. The dependencies and the compatibility of the individual structural elements have to be determined as well; typically while using a matrix approach. For IT software projects it might be possible now to guess the

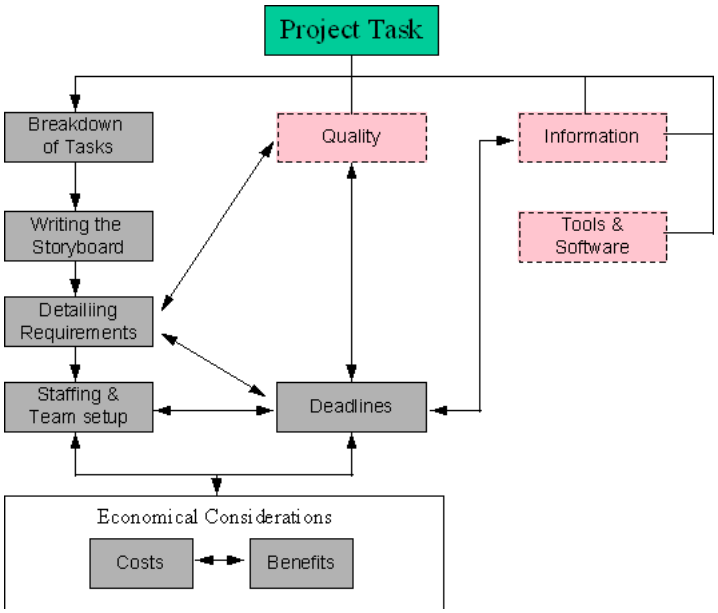

Figure 5.4: Internal and external dependencies of a project [24]

program structure, the interfaces, perhaps common subroutines and functions and also the required program classes. In fact, IT projects typically follow a *object-oriented top down-approach*. Figure [5.5] shows a top-down approach in order to evaluate the WBS.

After the WBS has been established initially, it is necessary to define particular tasks which can be assigned to individual project members, groups, or perhaps external providers:

- By construction, every task is self-confined and can be completed independently from each other (except for streamlining the interfaces).
- The task should be considered as 'black box' [A → task → A']:
  - A defined input [A] is provided to the [task],
  - the [task] has to react upon receipt in a deterministic manner as defined in the corresponding use-case, and
  - the [task] produces a well-defined output [A'] .

From the point of the assigned person or group, a task is a sub-project and requires a comparable level of project management, in particular regarding efforts, deadlines, and quality. Tasks packages can be assigned to Systems or Subsystems within the WBS as detailed in figure [5.6].

5.4 Project Organization Structure

In parallel with the WBS breakdown, the organization structure of the project can now be established. Now, the staff members can be organized

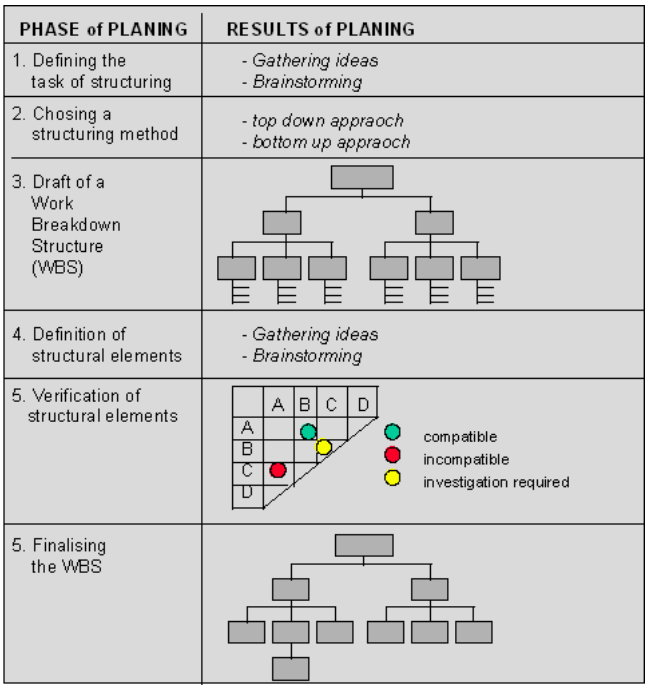

Figure 5.5: Top down approach for setting up the Work Breakdown Structure [24]

in groups while it is possible to estimate (at first glance) the size and the qualification of groups according to the WBS (figure [5.7]).

As a result, not only a (final) *organigram* and thus the resulting reporting chains is derived, but rather the assignment of the individual groups with tasks becomes transparent and perhaps allows to estimate possible incompleteness of tasks and staffing. This diagram shows in addition the Level of Responsibility for the individual groups and their co-operation in the whole project.

5.5 Scheduling: Assigning Deadlines and Resources to Tasks

Once evaluated, the WBS provides you with the layout or your project w.r.t. systems, subsystems, and tasks (as structural element). The top-down approach for the WBS is now complemented with a bottom-up approach to assign

- Deadlines
- Resources

to each activity which is known as schedule management.

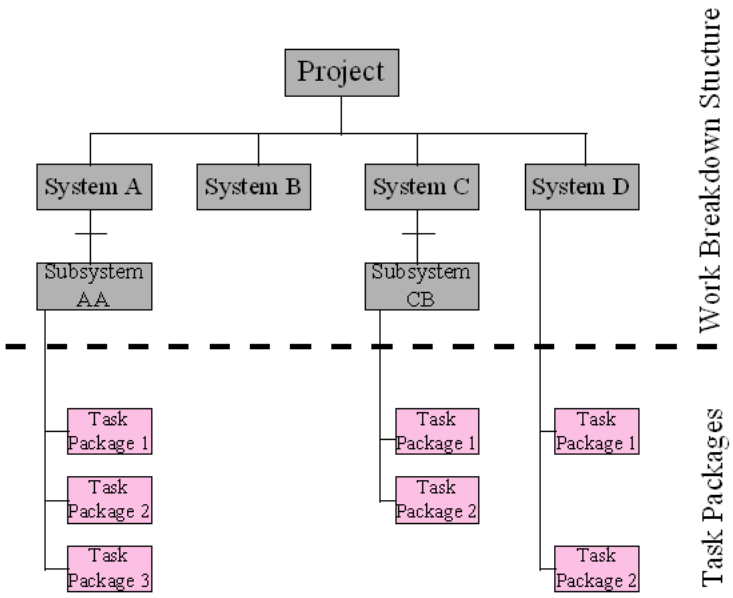

Figure 5.6: Assigning task packages after the breakdown of the project in a WBS [24]

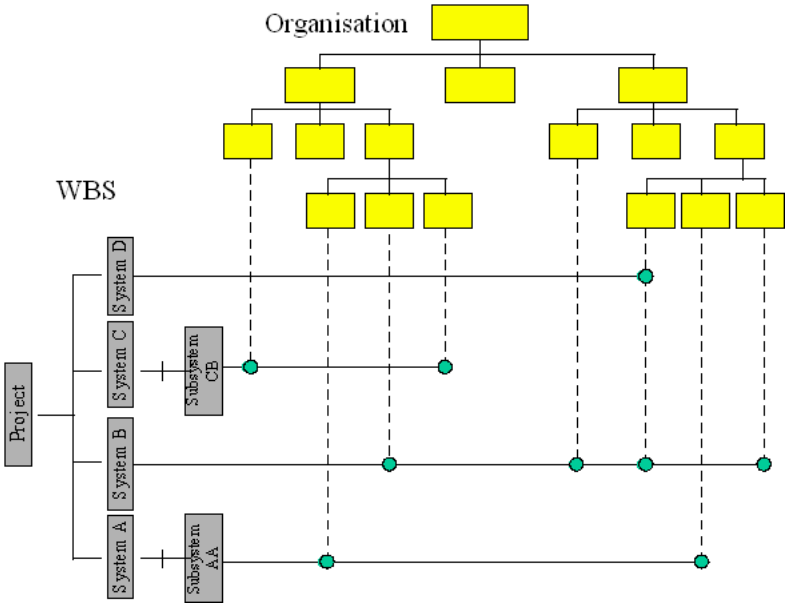

Figure 5.7: Correlation of task with organization [24]

5.5.1 Gantt Charts

This particular 'view' is typically visualized by a *Gantt* chart (figure [5.8]) (named after the inventory *Henry L. Gantt*). Here,

- the duration of the individual activities are displayed as *bars* which
- have a related start and end date and
- showing their (linear) dependencies.
- Steps can be combined to tasks, sections or phases.
- The level of completion or critical steps can be easily visualized.

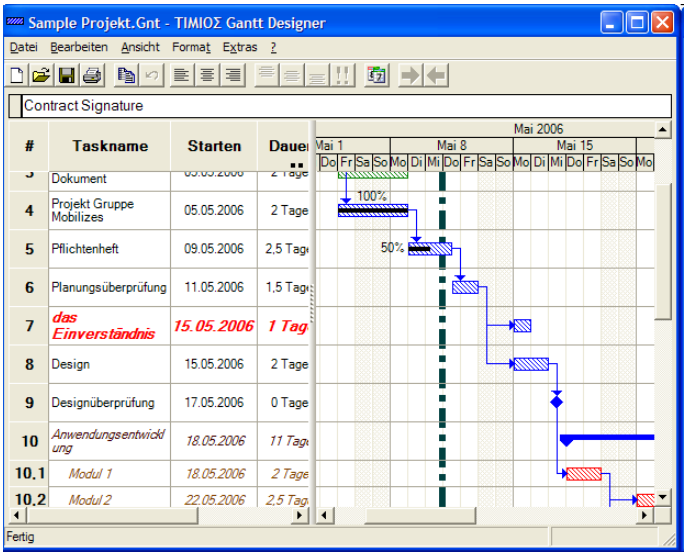

Figure 5.8: A snapshot view of the freeware Gantt program TIMIOS [44]

Typical computer programs to provide Gantt charts techniques, allow in addition to define

- common human resources (availability's) and
- available budgets for the whole project,
- which can be shared among the individual activities, according to definable keys (percentages).

A mathematical breakdown of those assigned numbers yields an estimate of

- the level of completeness regarding activities, section, and phase level,
- the used resources and budgets with the same level of detail.

Regarding the very advantages of Gantt charts (in particular using dedicated software, like Microsoft's Project), the disadvantages of Gantt charts result in the facts:

- Gantt charts do not show the functional dependencies (compared to the WBS).
- Larger projects result in very large Gantt diagrams and are hard to read.

5.5.2 Lists

The calculation within a Gantt chart is based on a spread-sheet; just the representation of the results follows the 'bar' scheme, known as Gantt chart or diagram. However, the basic calculation involved can be realized with basic spread-sheet means, or can be expressed as chained lists. Those simple spread sheets can be enhanced with inter-related sheets, where resources are defined can correlated with the individual activities. Though, very easy to use and to adopt to a project, the information here is very limited and only suited for smaller projects (figure [5.9]).

Activity	Name	Predecessor	Successor	Duration [Days]	Start Date	Due Date
1	Definition 2		2		2.2.2009	8.2.2009
2	Teambuilding	1	3	10	9.2.2009	28.2.2009
3	WBS	2	4			
4						

Figure 5.9: Simple list with a hierarchical breakdown of activities and schedule

5.5.3 Netplan Techniques

The *Netplan* technique is defined in `DIN 69 904` with the following scopes:

> "The netplan technique includes procedures for project planing and control. The netplan is a graphical representation of flow-structures, in order to visualize logical and temporal dependencies of activities."

There exist three types of the Netplan approach whether the activities are describes as *vectors* or *nodes*, or whether instead of activities events are meshed in nodes.

Figure [5.10] shows an activity *node netplan*. Here, one node displays additionally resources and duration of the respective activity.

5.5.4 Critical Path Analysis (CPA)

Once we have identified the resources required to achieve or complete a certain project task, we can gauge this task against all others. Gauging means, we have to consider

- the respective resources required, and
- the effective dependencies
- for every task.

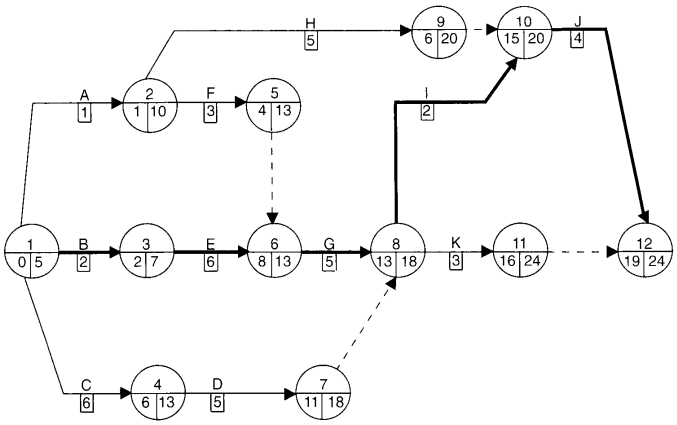

Figure 5.10: Sample for a Netplan with Activity-Node meshing [24]

In a well-behaving WBS, dependencies can be expressed simply as number of interfaces with respect to it's completion schedule. Figure [5.11] lists five milestones (10 to 50) and six activities (A to F). The 'criticality' is provided in term of completion time (`t`).

PERT chart

Thus the paths B+C are most critical for the project. This diagram is a so-called **PERT** chart (*Program Evaluation and Review Technique*), which is equivalent to the Netplan 'activity-vector' approach.

As part of the risk management, PM has to have at any time a good understanding of those tasks which are considered 'critical' and shall watch those very closely.

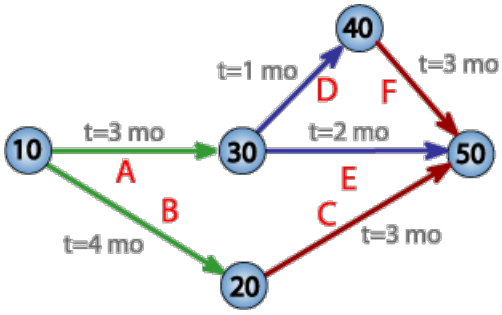

Figure 5.11: Critical Path Diagram [50]

5.6 IT Project's Phases

Until know we have described the first step of Project Management, thus *planning the project*. The planning phase can be broken into different stages:

1. Setting up a team which actually does the planning.

2. Creating a *Work Breakdown Structure* which describes the functional groups (broken down in systems, subsystems, and tasks) of the project and allows an estimate of dependencies and required efforts.

3. Creating a *Schedule* which tells, when the individual task have to be completed and in addition shows their execution dependencies.

4. *Assigning Resources* to the tasks in terms of man-power and budget. At the end of this step, a Masterplan for the project has been established which includes the above mentioned ingredients. Potentially, already now it becomes apparent which organizational changes to the project are essential, thus which teams need to be set-up initially and which long-lasting SPLs need to be assigned.
   What we are still lacking is

5. the determination of in particular *financial resources* to the project,

6. a dedicated *risk management* for the individual tasks.

A significant work during this Project Planning phase is to establish the required individual plans as part of the *Masterplan*. Unlike the documents provided during the *Project Initialization*, a fine-grained time-schedule and a task-schedule is the outcome of the project planing.

Product Planning and Development

Now, we need to turn our focus from *project planning* to *product planning*. We will see, that IT project management has some specific ideas how to set-up a *development* and *release plan* in order to meet the required product quality.

5.6.1 Software Development Lifecycle Model

Unlike general project management, IT projects – which are mainly software – projects follow a specific approach:

SDLC

Due to the imperfection of the product and a rapidly moving market and operation conditions, software needs to follow a *lifecycle* model. This has to be considered as well in the product design but additionally within the project management.

Thus, this software lifecycle spans from the design phase (as part of project planing),

- includes the coding (execution of the project),
- setting up a quality management for the development and product,
- the roll-out of the product and it's
- operation and maintenance.

Different from other project, software projects can substantially supported by software itself. Dedicated tools exist to measure and improve

product quality. However, as with most projects: the better the design the better the quality is and the faster the realization could be. Or, reversely: A poor and/or complicated design will probably lead to a mediocre product which requires substantial support and efforts during realization (coding/implementation) and operation.

According to our today's understanding, an IT project evolves in five *phases* under responsibility and control of a dedicated IT Project Management. Figure [5.12] explicitly makes a reference to an IT software project.

1. Initialization (Project idea)
2. Evaluation
3. Draft
4. Detailed Study
5. System design
6. Project finishing

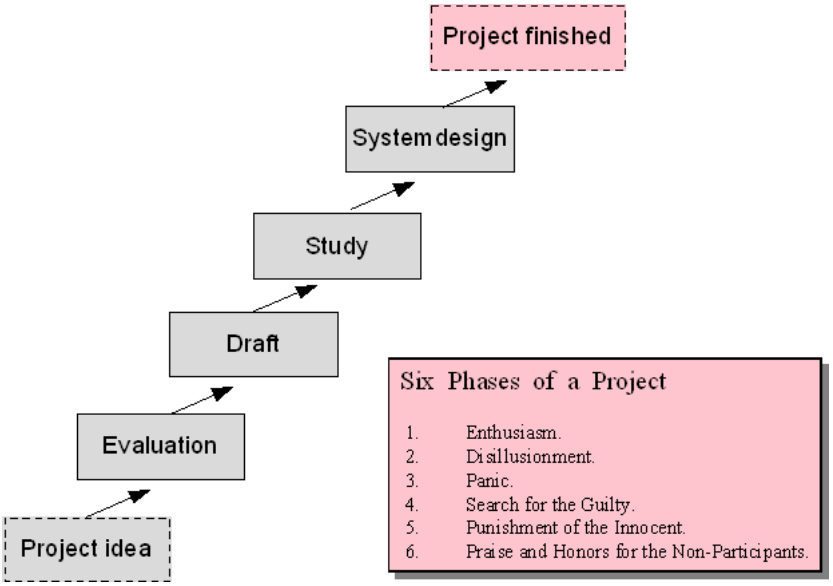

Figure 5.12: The phases of a IT project official and alternative

Considering the construction of special tailored software, it became soon aware that software never is delivered as expected by the sponsors:

- During the project evolution, the requirements of the software has changed; in particular because the deficiency of the original approach became apparent.

- The functional aspects of design were not met, thus additional development is necessary.
- The quality of the software is below expectation and during operation an uncontrolled (and perhaps unrecoverable) behaviour is been observed (commonly referred to as 'bugs').

Software evolves in time, which is know as *Software Lifecycle Model* (figure [5.13]:

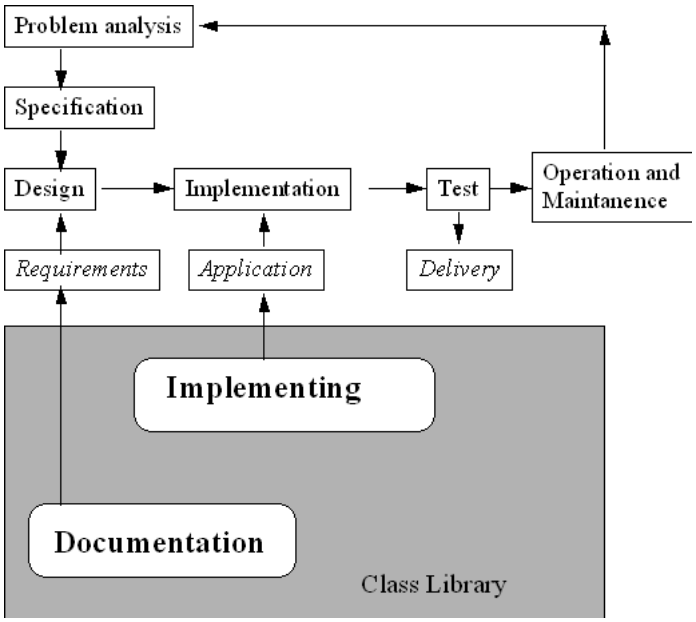

Figure 5.13: Software Lifecycle Model [45]

5.6.2 Spiral Model

More complex IT projects are not just focused on software development. Rather the project's tasks may include:

- Adoption and configuration of standard software.
- Considerable amount of special tailored software.
- Configuration of hardware, operating system, and middleware.
- Roll-Out of hard- and software.
- Introduction of the new solution into operation.
- Customization of the current environment to include the new solution.
- Education of the technical staff to become familiar with the new system.
- Promoting the new solution to external customers.
- Providing a full documentation for the integrated solution.

Thus, today's IT projects require a merge of

- special software development methods and
- adopt classical project management approaches.

Milestones

One way to combine both worlds is to break down the software specific methods in a Milestone approach in a so-called (Milestone) Spiral model:

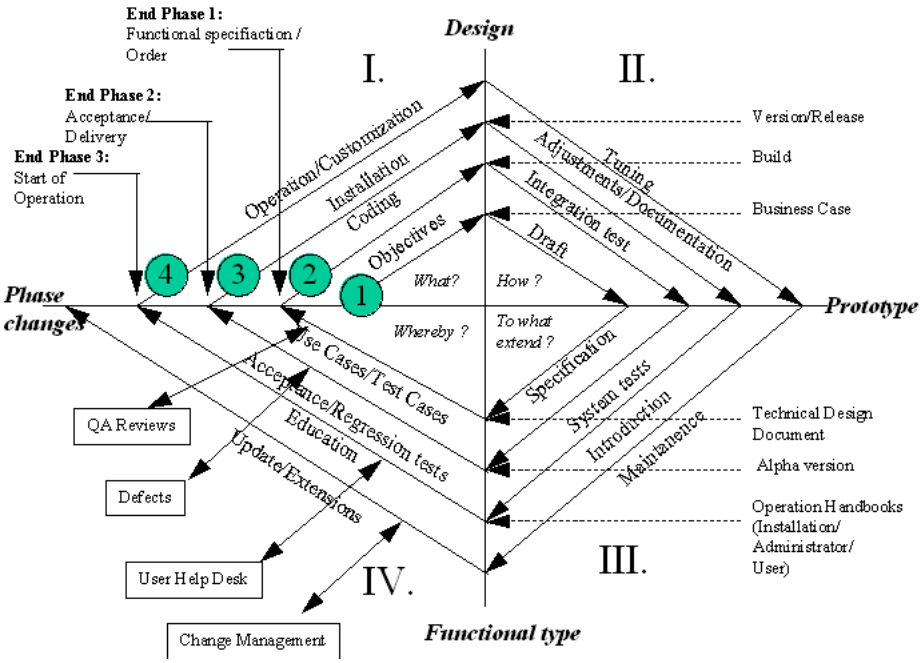

Figure 5.14: Spiral model for complex IT projects

The spiral model [5.14] is essentially a phase model and describes the software development in cycles, while determining the four essential phases:

- Design
- Development
- Functional testing
- Operation

Important here is, that the phase changes are well determined in terms of milestones to be achieved. Apart for development issues, the model also includes operational aspects. The four quadrants of the spiral model indicate in particular the basic questions needed to be raised:

- What needs to be done ?
- How do we achieve the results ?
- To what extend do meet our objectives ?

- What are the relevant means ?

The milestones are additionally clearly described:

- Start is the Business Case
- Next step is the technical design document
- The build of the first (working) version, tells that coding is on it's way
- The first alpha version will be delivered to **QA**, thus the product should be mainly completed.
- Now, the software enters the usual release/development cycle
- And finally, not only the software but also the documents are in place for production.

Unlike the **WBS** or the Gantt chart, the spiral model does not try to assess the logical structure of the project, nor tries it to pinpoint the project's current state or resource exhaustion. Rather, it tells the Project Manager what is in place and what is missing, independently from the **WBS**. In this respect, the spiral model provides a framework for software development and can be used as a ruler for IT project management.

5.7 Background, Exercises, Facts

- Task 1:
  Check for your favourite Software tool/operating (commercial) software the existence of Installation and Operating documents. How is the maintenance of the software facilitated ?
- Task 2:
  How does the current 'overall' availability of the Internet change the paradigm of enclosed documents and the update cycle ?
- Task 3:
  Discuss this current trend in terms of high-availability software. How does this trend potentially impact the way current software is designed and implemented ?
- Task 4: Do some Internet research regarding 'Extreme Programming' and compare this with the assumptions in this chapter.
- Task 5:
  What is the difference between a standard 'document' and a 'plan'?
  Is a structured document already a plan?
  What makes a 'plan' a 'plan' and what is the relationship with a 'project'?
- Task 6:
  Provide a brief summary of all relevant plans to be established during the planing phase !
- Task 7:
  Consider the risk of a project.
  Do some Internet research about 'Deepwater Horizon' (http://www.spiegel.de/thema/oelpest_im_golf_von_mexiko/).
  How to estimate the "impact of a risk"?
- Task 8:
  One relative new branch of statistics is the Bayes' probability.
  Could this approach help to provide a better risk measurement?

Part V

Standards and Frameworks

In this section, I will focus on a description of today's Project Management *frameworks*. While the *DIN 69 000* provides us with a *top down* view on Project Management

- regarding scope and definitions,
- the organizational structures,
- reporting channels, and
- particular tasks

become more and more detailed, they are complemented today by a new *bottom up* approach known as *Agile Project Management* and in particular *Scrum*.

Chapter 6

DIN Norm 69 000 for Project Management

6.1 Scope of DIN 69 000

It is a good tradition in Germany to formulate standards, rule-sets, and measurement methods in terms of the so-called '*Deutsche Industrie Norm*' DIN. One of the most famous DIN standards which caught some attention even in the US and Japan in the 70s of the last century was the standard DIN 45 500, defining measurement and quality requirements for High Fidelity (music) components.

DIN, EN, ISO

Most of those national German standards are meanwhile integrated and streamlined into the international standardisation scheme: The *European Norm* (**EN**) and the In*ternal Standardisation Organisation* (*ISO*); for example the DIN 45 500 has been replaced by DIN EN 61305.

Since regulations for quality are the scope of the German standardisations, there is no surprise, that the standard DIN 69 000 (now: ISO 10006 '*Guidelines for Quality Management in Project*') discusses issues of Project Management in six different parts:

1. DIN 69 900: Netplan Techniques
2. DIN 69 901: Project Management
3. DIN 69 902: Operating Resources
4. DIN 69 903: Costs and Activity Accounting, financial Resources
5. DIN 69 904: Project Management Systems
6. DIN 69 905: Project Operation

However, regarding IT Project Management, the standard ISO 10006 is complemented in particular by:

- ISO 9001/9002: Quality Management Systems
- ISO 10007: Guidelines for Configuration Management
- ISO 10012: Measuring Management Systems
- ISO 19011: Auditing Guidelines for Quality Management Systems

Thus, the main scope of ISO's Project Management standards defines the *requirements and means of a quality process.*

n particular it focuses on an effective project controlling, which is disentangled from project management.

6.2 Project Management according to DIN 69 901

The standard DIN 69 901 simply introduces in item 2 the definition of a project, project management, and project operation. Additionally, it provides in item 3 descriptive terms for the project organisation, in particular the reporting structures. Further, the structural elements of project tasks are laid out. Here, DIN 69 901 introduces '*Projektstrukurplan*' **PSP** (figure [6.1]) as equivalent to the *Work Breakdown Structure WBS* of **PMI**'s *PMBoK*.

PSP

The **PSP** can be laid out in accordance to

- the logical structure ('Aufbaustruktur'),
- the schedule of the tasks ('Ablaufstruktur', 'Netzplan'),
- the existing or chosen organisational structure ('Grundstruktur', 'Wahlstruktur'), and perhaps
- any mixed structure of the project.

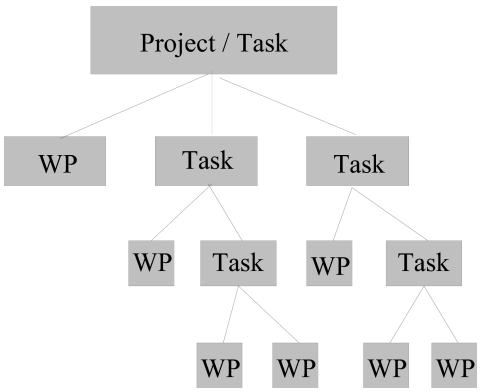

Figure 6.1: The 'Projektstrukturplan' according to DIN 69000

Item 4 introduces the terminology about *project organisation*, *project management*, and the rôle of the *sub project leader* ('*Fachprojektleiter*').

Governance

Finally, item 5 describes the necessary terms for *project governance*:

- Goal of the project,
- definition of the project,
- product specification,
- project information (system), report, final report, project documentation,
- limits, and
- completion ratio.

Of course, one might expect that now the often used German terms

Pflichtenheft, Lastenheft

- 'Lastenheft' (requirement specification) and
- 'Pflichtenheft' (technical specification)

would have been explained. However, this is suspended and finally defined in `DIN 69 905`. The 'Lastenheft' and 'Pflichtenheft' identify the attributes of the project's subject: the product.

The commercial conditions among the involved parties needs to be additionally specified in a

- 'Projektvertrag' (contract specifications)

detailing the costs and pricing schemes, penalties and procedures which are related to the project itself.

6.3 Operating a Project (`DIN 69 905`)

According to my understanding, `DIN 69 905` introduces explanations for two separate PM issues, which can be categorizes as mainly

- commercial aspect and
- specific project aspects.

While `DIN 69 905` provides a specific view on the project (internal) terminology, which is complementary to the PRINCE2 or the PMBoK approach, the introduced commercial framework is of great importance, since it's impact is often underestimated.

Commercial framework

- Bidding:
  Typically, the commercial life-cycle of a project initiated by the Sponsor (*Auftraggeber*) starts with a Offer (*Angebot*) which shall include a List of Requirements (*Anforderungskatalog*) mandatory for the actual project. However,
- makes no clear distinction between
  - Request for Information

RFI

  - Request for Tender RFT
  - Request for Quotation RFQ
  - Request for Proposal RFQ

In case, the *Sponsor* and the *Tender* are already coupled by some contract, the Sponser may issue directly a **RFP**. Otherwise several Tenders may submit their Proposal during the (well determined) *Bidding period*. The Sponsor will finally make an *Appointment* (Zuschlag) and will chose a specific Tender to realise the project and release a *commercially binding Order* (Bestellung).

- Contracting:
  At that point, Sponsor and Contractor define the particular conditions of the project. Here, two major documents need attention:

Lastenheft
  - The *Contract Specification* (Lastenheft) details the set of requirements of the Sponsor regarding the Deliveries (Lieferungen) and the Attainments (Leistungen) of the Contractor during the contract phase.

Pflichtenheft
  - The *Functional Specification* (Pflichtenheft) includes the Proposal of the Contractor, how to realise (in schedule and quality) the contract conditions as laid down in the Contract Specification.

- Execution:
  DIN 69 905 provides a rich set of project management terms, including Project Organisation, Project Management Systems/Tools, Risk management, and many others.
- Suspension:
  A project could conditionally brought on hold by the Sponsor; thus it may be suspended for a while (Sistierung). This condition should be included in the project Contract, because it will certainly produce additional costs for the Contractor.
- Acceptance:
  The project as a whole or in parts can be Commissioned by the Contractor and Delivered to the Sponsor. It is the duty of the Sponsor to start a (defined) Approval Process and upon success, to notify the Acceptance of the Delivery (Approval Report) which requires mutual consensus. The Sponsor may also only partly express Acceptance, or – if things go badly – *Refrain Acceptance*.
- Warranty:
  DIN 69 905 considers the Warranty (Gewährleistungsphase) as particular project phase. During this phase, the Contractor has the obligation to provide additional warranty services, which may be additionally charged.
  Distinct from the Warranty is *Accommodation* (Kulanz), which may

include unsolicited (freiwillige) services by the Contractor; whether charged or uncharged.

- Closure:
  The project ends with it's formal Closure. Mandatory is a project Closure Report. A Closure will either happen in case the project is successfully finished, or it is conditionally or unconditionally cancelled.

6.4 Budgeting Projects and Controlling Costs

The standard `DIN 69 903` provides the terminology to

- provide a budgeting of a project (financial sources)
- allow the control of costs of the project (financial sinks).

Cost Breakdown Structure

The flow of the financial means (the Expenditure) during the project's execution can be best controlled by means of a cost planing and a particular *Cost Breakdown Structure* (Kostenstrukturplan). The effective costs have to be balanced by the project's *Attainments*, thus both need to be gauged for a specific Working Package or Task, which is will typically accomplished by the Project Controlling (figure [6.2]).

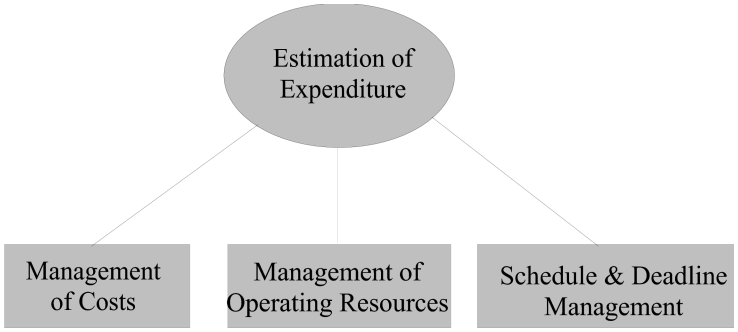

Figure 6.2: Elements of Budget Planing and Cost Control

6.5 Project Management Systems

PMS

The standard `DIN 69 904` tries to lay out the qualifications of a *Project Management System* in terms of a raw model. The main requirements of a PM System are (figure [6.3]):

- *Flexibility* – the PM Systems need to be adoptable for new or changing conditions.
- *Universality* – the system should support different usage conditions.
- *Modularity* – the system is constructed out of several subsystems to be used as a toolbox.

- *Compatibility* – the subsystems provide standardised interfaces to allow a systematical aggregation of functions.
- *Transparency* – the systems allows to visualise the structure and the dependency of the project and it's tasks.
- *Prevention* – the system support the approach 'prevention instead of reaction'.

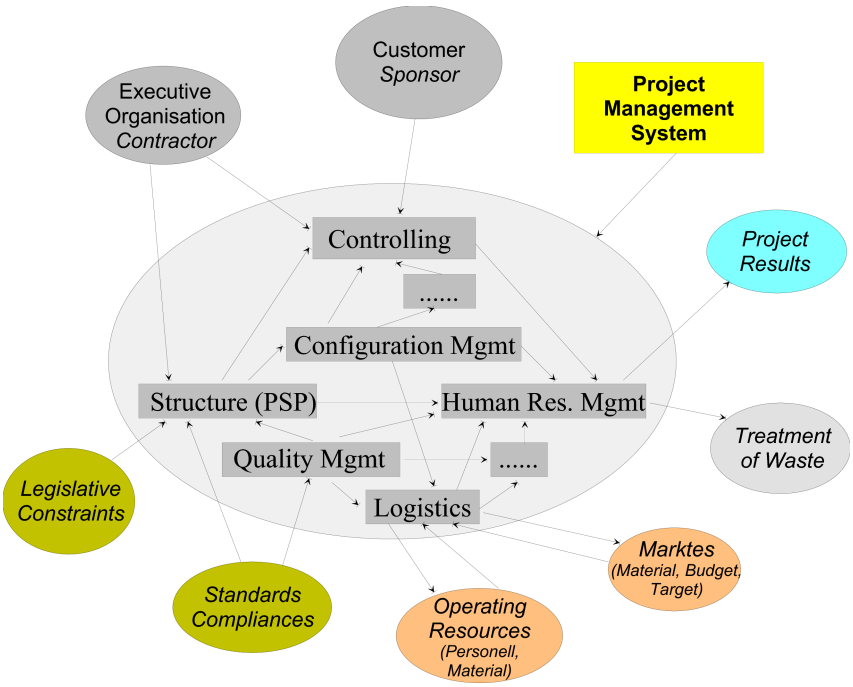

Figure 6.3: Bird's view of the structures and processes supported by a PM System

WithinDIN 69 904 the elements of project management are mentioned. PM elements are assigned specific functions of a Project Management System and can be used as a toolbox.

6.6 Netplan Techniques

The Netplan Technique introduces a schematical/graphical view of the project with respect to its tasks, dependencies, and schedule. Unlike the *Gantt* approach, it provides a vectorized presentation and allows a multi-level dependence of tasks. Similar to the *Critical Path Analysis* **CPA** the timelines can be visualized and the criticality can be determined.

Essentially, Netplan consists of

- tasks or events, represented as rectangle or circle,

- activities (displayed as arrows), attached to tasks and events,
- logical junctions of activities, and perhaps
- additional hints linked to tasks and/or activities.

The strength of Netplan originates from the set of coherent attributes assigned to tasks and activities. Thus, it is possible to clearly follow the project's flow.

Beginning and ending task are visualised as specifically. The drawings of a Netplan may become complex and in particular large. Therefore, it is possible to split the drawing on individual pages while clearly assigning the relationships in terms of successor and predecessor.

6.7 Project Controlling

One of the most important implicit impacts for Project Management originating from `DIN 69 000` is to focus on Project Controlling. One of the lessons learned from Cologne's Subway construction disaster, is the missing controlling of the Sponsor.

Typically, it is task of the Project Management to achieve controlling on there on behalf. However large or 'Multiprojects' require an additional level of Project Controlling, since individual decisions within a sub-project might influence the whole project substantially.

Thus we can identify Controlling with respect to the following tasks (figure [6.4]):

- Operative Controlling
- Financial Controlling
- Systemic Controlling

Controlling is a shared task between the Contractor and the Sponsor. For large project it might the feasible to assign a third party to be responsible for Project Controlling.

Controlling = shared task

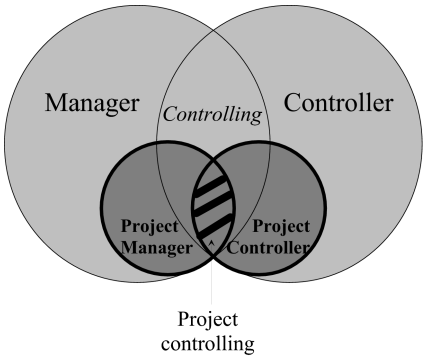

Figure 6.4: Shared responsibility for Project Controlling [4]

Chapter 7

PRINCE2 – Projects in Controlled Environments

From the discussion of the former chapter, we have seen that projects can be subdivided into several legs, or using the PRINCE2 terminology stages, generally understood as *project life cycle*:

- Project Specification and Design
- Project Execution (IT: Development/Coding + Testing)
- Project Roll-Out (IT: Release Management, Operation support)

The British PM de-facto standard PRINCE2 actually widens this view and complements it with the *product life cycle*:

- Product Initiation: Idea, Trigger, Feasibility
- Product Design: Study, Layout
- Product Realisation: Implementation
- Product Operation: Use the Product
- Product Termination: Scrap the Product

I will now introduce the PRINCE2 framework and to identify it's specifics for Project Management.

7.1 Origin and Scope of PRINCE2

PRINCE has been developed as project management method in 1989 by the British Central Computer and Technology Agency (CCTA). The current PRINCE standard was published in 1996 as book 'Managing Successful Projects with PRINCE2'. PRINCE2 official home-page is hosted by the Office of Government Commerce (OGC). CCTA OGC

In order to fully apply PRINCE2 for IT Project Management, the organisation running the project should follow the **ITIL** (*IT Infrastructure* ITIL

Library) approach, also defined and provided the OGC.

According the it's home-page, PRINCE2 has the following scope:

> "PRINCE2 is a process approach to project management, fitting each process into a framework of essential components which need to be applied throughout the project. PRINCE2 helps you work out what roles should be involved in your projects, what they will be responsible for and when they are likely to be needed. The set of processes and controls provided give you the structure that will support the life of the project, and explains what information you should be gathering along the way. The PRINCE2 method demonstrates how your project can be divided into manageable chunks or stages, allowing you to plan ahead more realistically, and to call on your resources at the time they are most needed." "PRINCE2 acts as a common language between all of customers, users and suppliers, bringing these parties together on the Project Board. And although PRINCE2 doesn't include contract management as such, it provides the necessary controls and boundaries needed for everybody to work together within the limits of any relevant contracts. In addition, the Project Board provides support to the project manager in making key decisions."

In addition, the OGC tries to convince potential users of PRINCE2 by declaring the following benefits:

> "PRINCE2's formal recognition of responsibilities within a project, together with its focus on what a project is to deliver (the why, when and for whom) provides your organisation's projects with:
>
> - A common, consistent approach
> - A controlled and organised start, middle and end
> - Regular reviews of progress against plan
> - Assurance that the project continues to have a business justification
> - Flexible decision points
> - Management control of any deviations from the plan
> - The involvement of management and stakeholders at the right time and place during the project
> - Good communication channels between the project, project management, and the rest of the organisation
> - A means of capturing and sharing lessons learned
> - A route to increasing the project management skills and competences of the organisation's staff at all levels."

In other words, PRINCE2 tries to 'glue' the different participants of a project and focuses on a common understanding ([7.1]) about the state of the project, while the actual communication can be standardized with predefined to-do lists and reports.

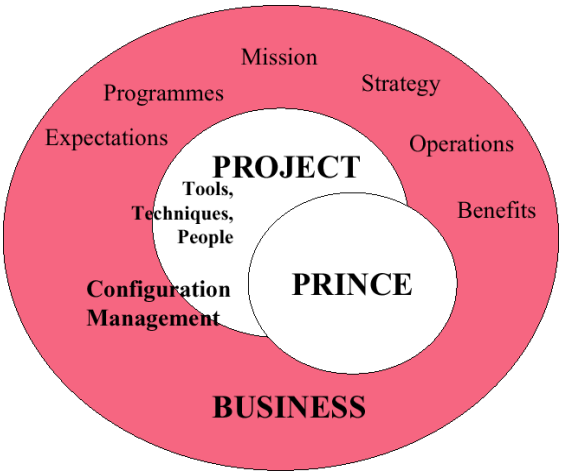

Figure 7.1: PRINCE relationship with projects and business [31]

7.2 The PRINCE2 Management Components

PRINCE2 subdivides management into eight distinct disciplines as shown in figure [7.2]:

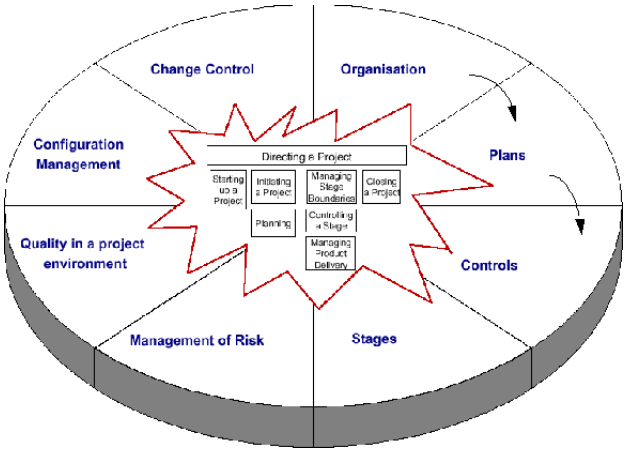

Figure 7.2: PRINCE2 Management disciplines [31]

A key understanding of the PRINCE2 approach is, that the disciplines deliver *Project Management Products* (**PMP**) to each other. Each PMP is identified, defined, and the delivery is controlled. The corresponding process flow, regarding responsibilities, decision-making, and support requirements are essential part of the PRINCE2 framework. In this sense, PRINCE2 sees itself as product-based planning technique. Since Quality Management is an essential part, it is believed that any Management Product (but not necessarily the final 'product' of the project) possesses an intrinsic quality.

PMP

7.2.1 Organisation

Customer/ Supplier

The Project Organisation (PO) describes in the PRINCE2 approach a Customer/Supplier relationship, independent whether they are part of the same organisation or not:

- The *Customer* will define the project's outcome and it's quality, while
- the *Supplier* provides the resources and skills to generate the outcome.

The PO has to be established independently from the 'line operation' and is instantiated over the project's lifetime. The generic project rôles for PRINCE2 defining the management responsibilities can be picked up from figure [7.3]:

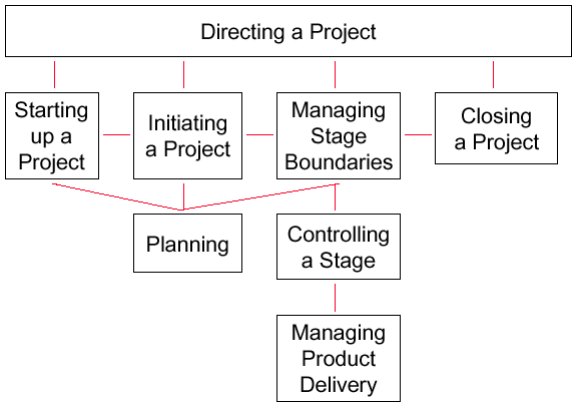

Figure 7.3: PRINCE2 Project Management structure [31]

Within this model, four management layers are defined :

- Direction of the project (Supervision)
- Day-to-day management of the project (Execution)
- Team Management (Staff)
- Team Members (work force)

Project Management Team

The first three layers define the PRINCE Project Management Team. Since PRINCE2 employs a Customer/Supplier relationship model, it's manifestation is facilitated in a '*Project Board*' (figure [7.4]) as joint forum:

Complex projects may be grouped together in a Programme. PRINCE2 defines a Programme as:

> "A portfolio of projects selected, planned and managed in a co-ordinated way and which together achieve a set of defined business objectives. Programme management methods and techniques may also be applied to a set of otherwise unrelated projects bounded by a business cycle.'"

Programmes may require additional support structures resulting in the following comprehensive PO structure [7.5]:

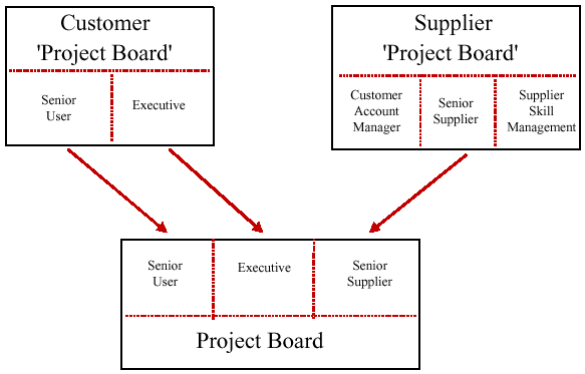

Figure 7.4: Customer/Supplier Project Management Organisation [31]

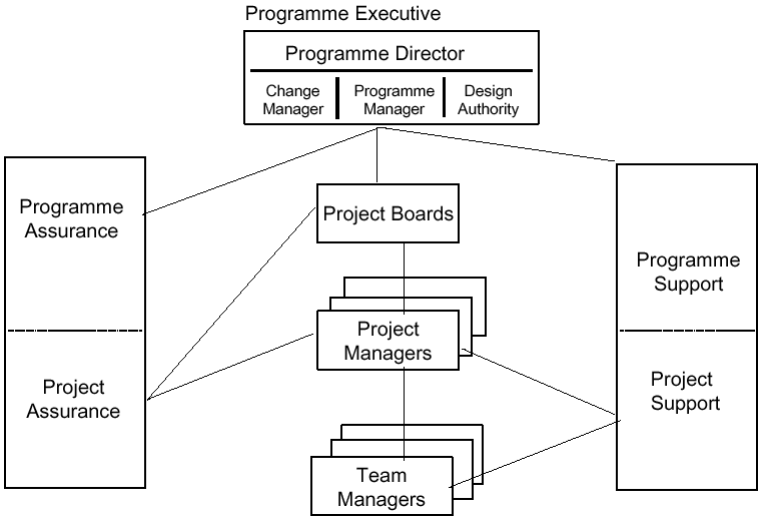

Figure 7.5: Programme Organisation [31]

7.2.2 Plans

In the PRINCE2 framework, a plan is a structured document, describing

- how,
- when, and
- by whom

a specific target or sets of targets are achieved, including time-scales, costs, and quality for a deliverable and need approval and commitment by the *Project Management Team* and additional approval by the *Project Board*. Approval

Plans are presented as management reports. While the format of a plan (ie. item-lists, charts, diagrams) are essentially free to chose, the plan should include the following components (figure 42): Plan components

- Produced products (deliverables)

- Activities to create deliverables
- Activities to validate the deliverable's quality
- Required resources and time for the above activities, staffing and skills
- Dependencies between activities
- External dependencies regarding information, products, and/or services needed
- Time schedule for the activities
- Monitoring points for the activities' progress

Typically, a Stage Plan includes a Summary Plan for the Project Board an a *Detail Plan* for control and daily work [7.6].

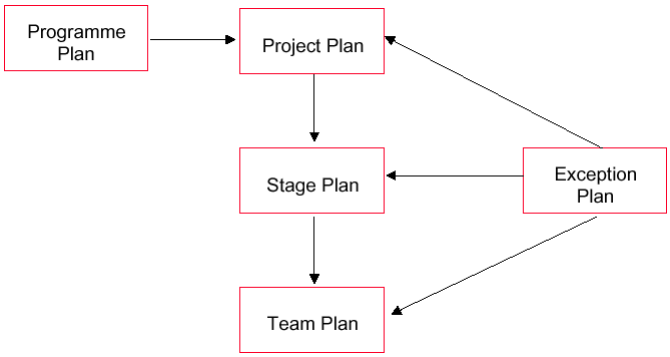

Figure 7.6: Components of a Plan [31]

If several projects are organised in a Programme, plans have to be carried out at different levels of the project as shown in figure [7.7]:

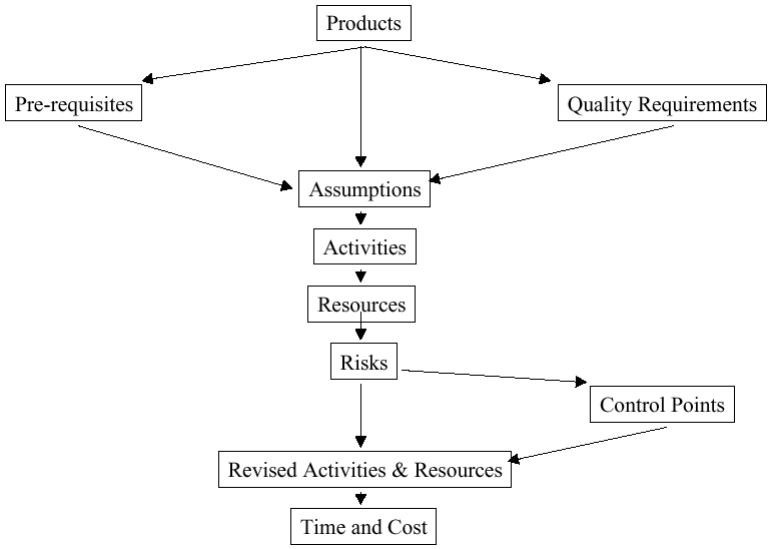

Figure 7.7: PRINCE2 Plan Level [31]

7.2.3 Controls

Controlling in the PRINCE2 framework means essentially to

- ensure that the project generates a product which meets the defined acceptance criteria
- ensures, that the project progress happens in time and within the resource and costs limits
- allow the project to be viable against the Business Case.

According to the layered management model, Control is responsible to the upper level, in order

- to monitor progress,
- compare the achievements with the plan,
- reviewing the plans,
- detect problems,
- initiate corrective actions,
- authorise additional work.

Deming Cycle

The PRINCE2 'control loop' approach follows essentially the *Deming Cycle* **Plan → Do → Check → Act**. The results of the major control points are important for Project Board in order to support the following decisions:

- Project Initiation
  Should the project be started?
- End Stage Assessment
  Has the stage been successfully completed?
  Is the Business Case still valid? Risks under control?
- Highlight Reports
  Regular progress reports
- Exception Reports
  Early warnings in case of problems and new substantial risks.
- Mid Stage Assessment
  Standard correction actions in case of forecast deviation.
- Project Closure
  Project finished as expected?
  Follow-on actions?
  Lessons-learned sessions?

7.2.4 Stages

Stages versus Phases

PRINCE2 defines a stage as 'a collection of activities and products whose delivery is managed as a unit' and is effectively a 'unit of work' carried out by the project team. In this respect, a *stage* is a partition of the project, unlike a *phase* which characterises a partition of the product life cycle.

Stage are an indispensable part of any project, since they allow to

- define the decision and review points
- adjust the precision of the forthcoming planning, and to
- improve scalability of the project.

Two distinct understandings of stages are possible in the same project:

- *Technical stages* are defined by a particular technical (production) method involved.
- *Management stages* identify intervals, which include commitments of resources and authority to spend.

PRINCE2 typically uses 'Management stages' for planning and control (figure [7.8]) which may include different technical stages:

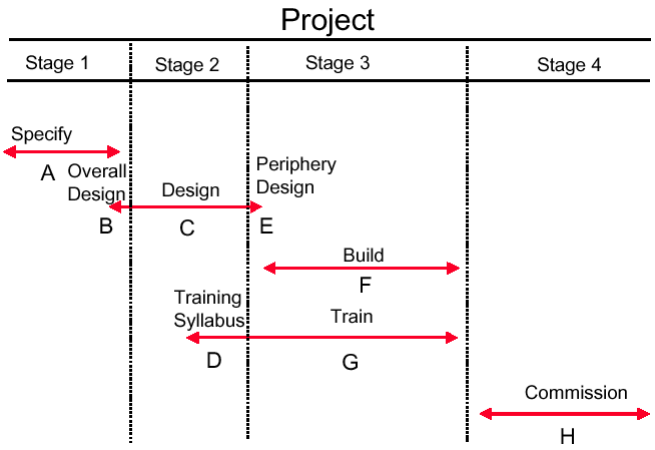

Figure 7.8: Breakdown of technical stages (A-H) wrt. management stages (1 to 4)

7.2.5 Management of Risk

PRINCE2 defines as risk as

> "The chance of exposure to the adverse consequences of future events".

and treats risks a major factor to be considered in the management of a project.

From the point of PRINCE2, risks can be categorised mainly into two types:

Business Risks

1. Business risks
   like validity and viability of the Business Case, alignment with future business strategies, political and legislative changes/requirements and environmental issues, customer acceptance and others.

2. Project risks — Project Risks
   supplier issues regarding third party components, organisational and inter-human factors, project-special issues including it's complexity and challenge.

Risk management is the responsibility of the Project Board and the Project Manager. In general, risk management follows a detailed risk analysis. Since risks can not be avoided, in case they happen, their impact has to be reduced as shown in figure [7.9].

Figure 7.9: Duties of Managers for Risk Management [31]

Risk management has to happen continuously in the project as part of the general reporting procedures (figure [7.10]).

7.2.6 Quality in a Project Environment

PRINCE2 picks up the quality definition from `ISO 8402`

> "Quality is the totality of characteristics of an entity which bear on its ability to satisfy stated and implied needs."

and requires the following *Quality Management elements*:

- Quality System — QS
  This is an organisation structure, the procedures and processes to implement quality management, either provided by the supplier or by the customer or both.
- Quality Assurance — QA
  An organisational unit setting up the **QS**, operating, auditing and maintaining it. **QA** can be realised within the project team or outside, e.g. commonly used in a programme.
- Quality Planing — QP
  Here, the objectives, requirements, and actions for the **QS** are defined. In the *Project Initiation Document* is should be explicitly provided as *Project Quality Plan*.
- Quality Control — QC
  Defines the process of controlling, i.e. examining a product whether it meets the defined quality objectives.

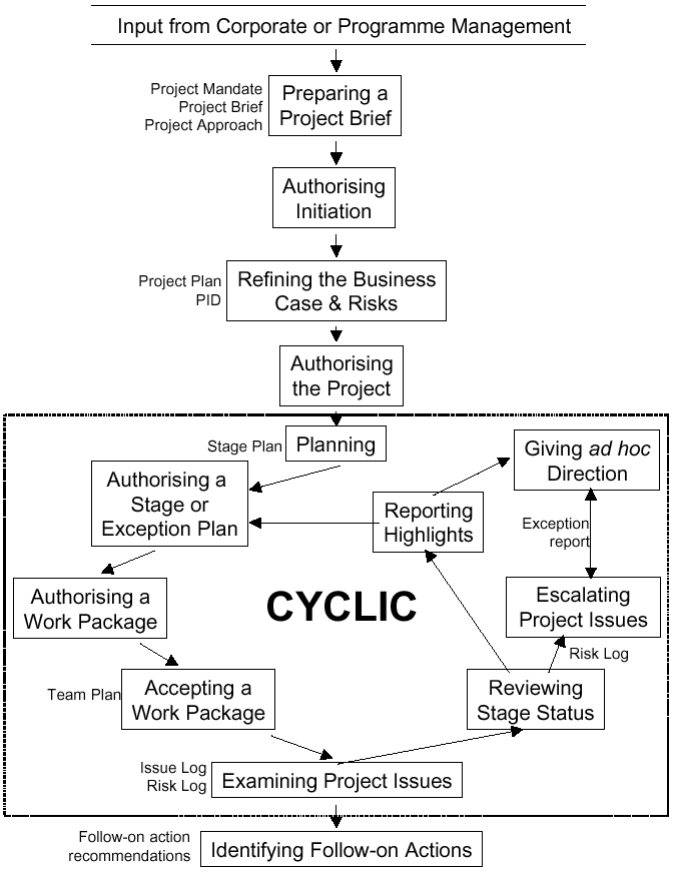

Figure 7.10: Risk flow and key points for management intervention [31]

ISO 9001

A particular Quality Management System is defined in `ISO 9001` . Though conformance with this standard can not be guaranteed by PRINCE2, running projects with the PRINCE2 method, with the path to quality as defined in figure [7.11], certainly support these requirements.

7.2.7 Configuration Management

Within the scope of PRINCE2, Configuration Management has to take about all the project's deliverables and the documentation, thus it has to identify, track, and protect the project's products and the responsibility of a *Librarian*.

Project Library

Configuration Management consists of the basic functions:

Planning

- Planning
  Defining the level of coverage for Configuration Management and how it can be achieved.

Identification

- Identification

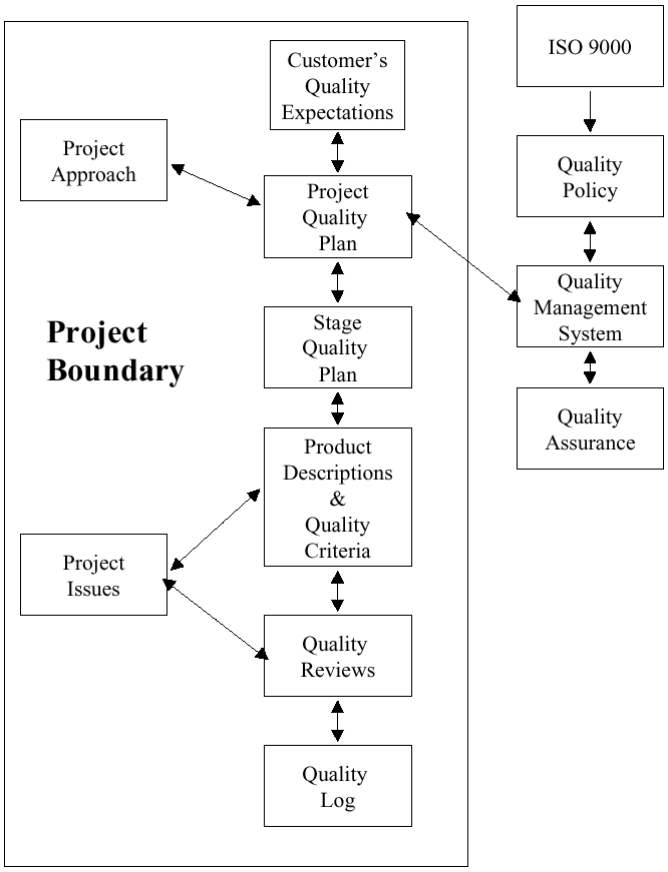

Figure 7.11: PRINCE2 path to Quality [31]

Detailing the components which are subject for Configuration Management.

- Control
  Ability to freeze a state of a product. After a freeze a certain authorisation is required to change the product.
- Status Accounting
  Records the status of the products.
- Verification
  Reviewing the actual state of a product wrt. the Configuration Management records.

Control

Status Accounting

Verification

In summary, PRINCE2 Configuration Management is comparable the same module in **ITIL**, however is specifically adopted to project management.

7.2.8 Change Control

Change Control within PRINCE2 has the following two main tasks:

Defining the level of Authority required to approve a particular change in the product. Verifying the Integrity of a Change: Conformance with the Business Case and whether it is beneficial. File a Risk Log. Considering the balance of time/cost/risk wrt. the foreseen change. Change Control and Configuration Management depend on each other, and it should use the established Configuration Management tools.

7.3 PRINCE2 Processes

7.3.1 Process Model

PRINCE2 defines eight major processes (figure [7.12]), including each a collection of sub-processes (figure 7.13]):

SU
- Starting up a Project
  Gathering basic information

IP
- Initiating a Project
  Getting agreement that we know what we are doing

CS
- Controlling a Stage and Managing Product Delivery
  Controlling development

SB
- Managing Stage Boundaries
  Taking stock and getting ready for the next part of the project

PL
- Planning
  Common planning steps

DP
- Directing a Project
  Senior management taking decisions at key points of the project

CP
- Closing a Project
  Making sure the project has done the job

7.3.2 Starting up a Project (SU)

Start Up

The *Start Up* of a Project has to be accompanied by the processes (figure [7.14]):

- Business Plan the determination of the basic business requirements triggering the project
- Project Board the identification of responsibilities, thus establishing the Project Board, and appointing the Project Manager (SU1)
- Project Management Team Design designing the project management team under consideration of the concerned parties (SU2)
- Appointing Project Manager Team the appointment of the PM team members (SU3)

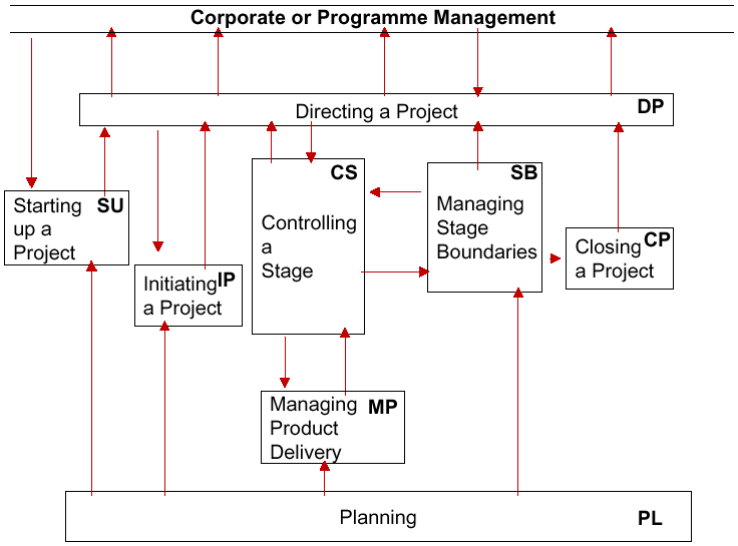

Figure 7.12: PRINCE2 processes structure [31]

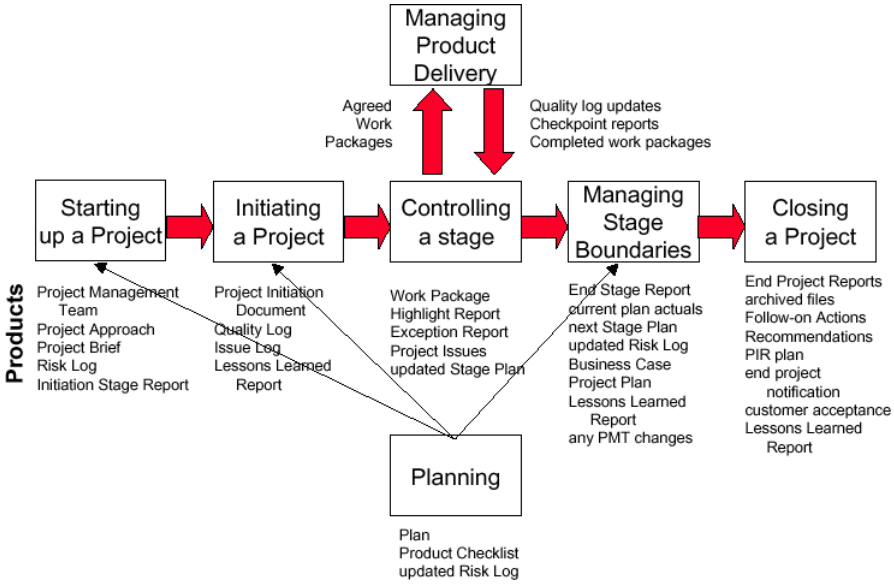

Figure 7.13: Sub-processes for the PRINCE2 main processes [31]

- Project Brief the knowledge of certain base information about the commissioning of the project (SU4)
- Project Approach the definition of the project Approach (SU5)
- Initial Stage Plan the creation of an *Initial Stage Plan* to enter the Initiation stage (SU6).

An important issue is also the Scalability of the project and thus in what context (Programme) it runs.

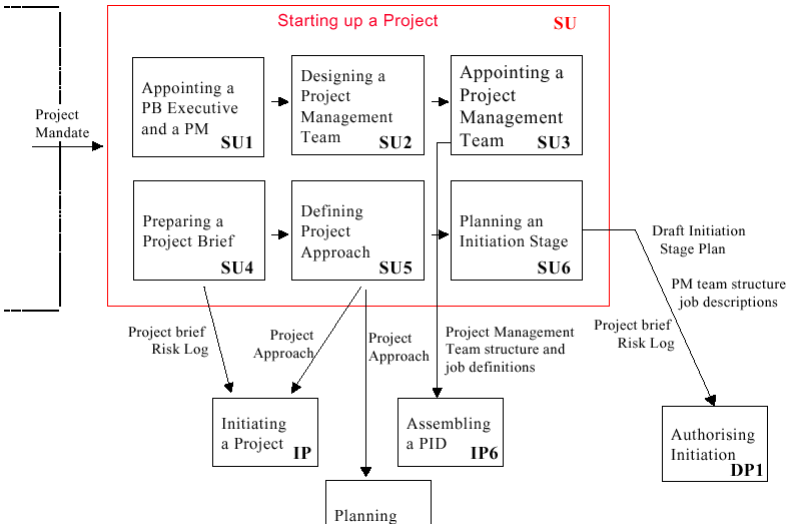

Figure 7.14: Starting Up Processes and Subprocesses [31]

7.3.3 Initiating a Project (IP)

Any successful project is determined by it's start and estimated end. The project's objectives and in particular the Business should be clearly understood by all participants. The solution path and the responsibilities should be determined in the first place.

Under these conditions, PRINCE2 defines the following *subprocesses* while Initiating a Project (figure [7.15]):

IP1 • Planning Quality

IP2 • Planning a Project
major products, activities, and risks; estimate efforts and resources needed; determine timescale

IP3 • Refining the Business Case and Risks

IP4 • Setting up Project Control

IP5 • Setting up Project Files

IP6 • Assembling a Project Initialisation Document

7.3.4 Directing a Project (DP)

The entitled project management has the authority to

- define the requirements for the project
- authorising funds
- committing resources
- make decisions on any changes requested by Project Management
- make decisions on exception situations

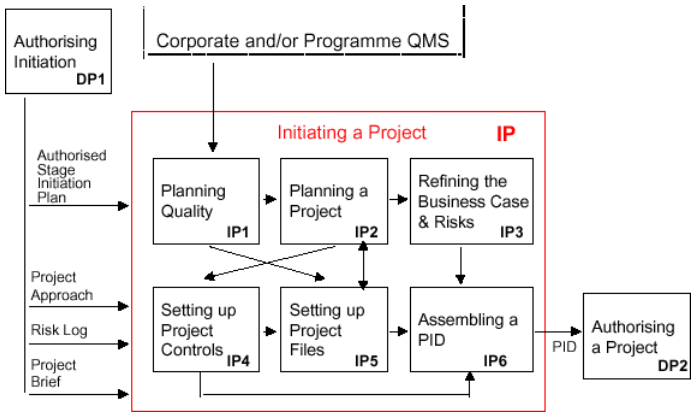

Figure 7.15: Initiating a Project [31]

- communicating with external stakeholders.

Directing a Project includes the following ([7.16]):

- Authorising Initiation DP1
- Authorising a Project DP2
- Authorising a Stage or Exception Plan DP3
- Providing Ad hod Direction DP4
- Confirming Project Closure DP5

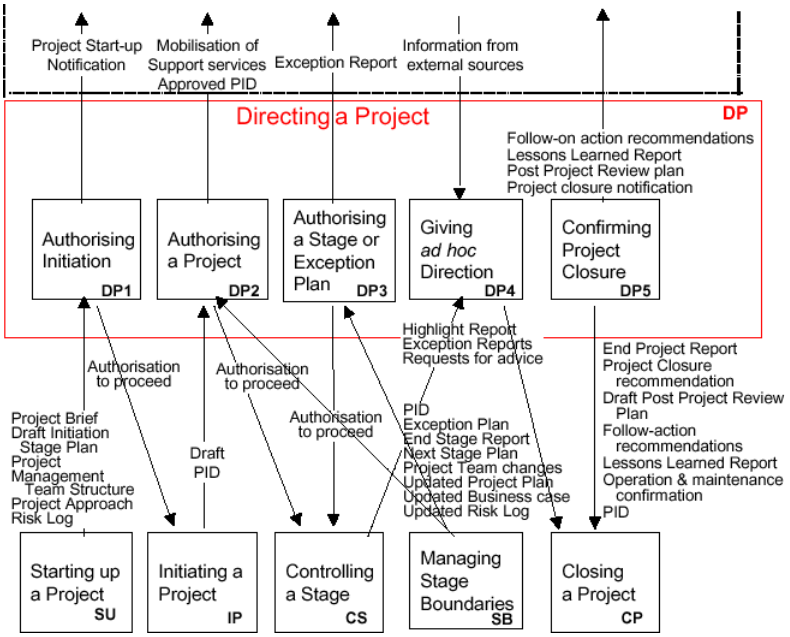

Figure 7.16: Directing a Project [31]

7.3.5 Controlling a Stage (CS)

Controlling a Stage is the most fundamental discipline for the actual execution of Project Management. At the end of every stage, a 'delivery' is assumed. In order to successfully provide this, management must focus it's attention

- on the realisation of the *delivery* or outcome,
- the *used resources* from the beginning to the end of the stage
- apply *risk control*
- keep the stage aligned with the *Business Case* and to
- monitor deviations from the initial plan (*loss of focus*).

The following sub processes are essential Controlling a Stage (figure [7.17]):

CS1 • Authorising Work Packages
CS2 • Assessing Progress
CS3 • Capturing Project Issues
CS4 • Examining Project Issues
CS5 • Reviewing Stage Status
CS6 • Reporting Highlights
CS7 • Taking Corrective Actions
CS8 • Escalating Project Issues
CS9 • Receiving Completed Work Package

7.3.6 Managing Product Delivery (MP)

Managing Product Delivery has two directions:

- Third Party Products may be needed to be incorporated into the Project.
- The (Sub-)Project has to deliver Products (defined as Work Package) to the Project or the Programme for Integration.

In the last case, it is the responsibility of the Team Manager to ensure, that planned Products are created and delivered by the team to the Project.

In order to streamline the delivery, the following sub processes are required ([7.18]):

MP1 • Accepting a Work Package
MP2 • Executing a Work Package
MP3 • Delivering a Work Package

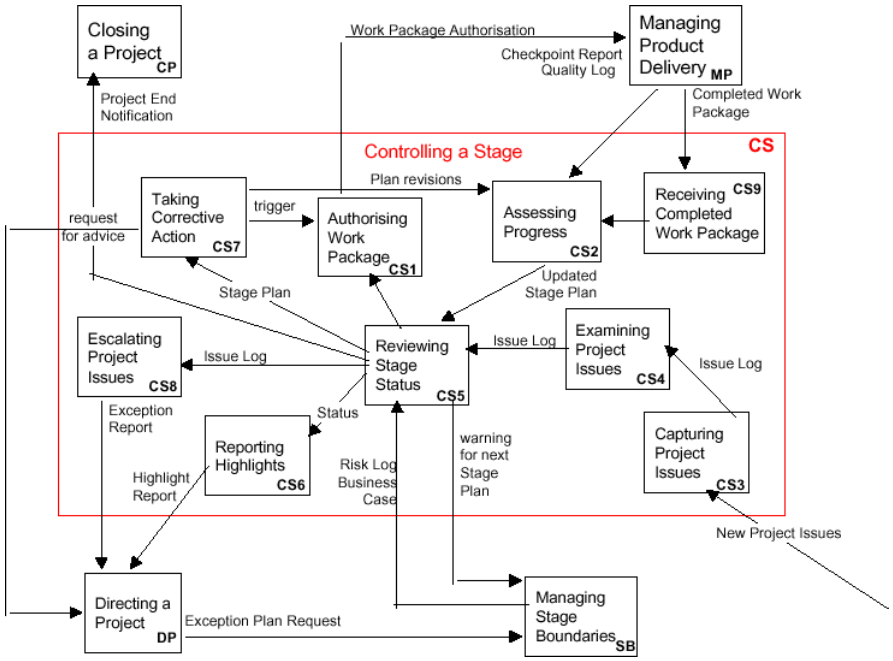

Figure 7.17: Controlling a Stage [31]

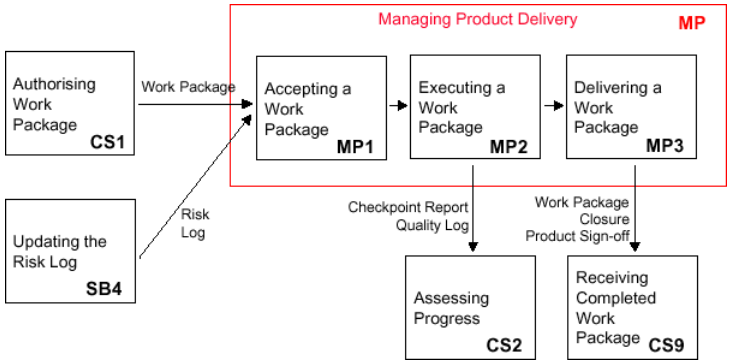

Figure 7.18: Managing Product Delivery [31]

7.3.7 Managing Stage Boundaries (SB)

Transitions between stages happen regularly during the progress of the project. The process Managing Stage Boundaries has the tasks (figure [7.19])

- to assure the Project Board that all Products in the current Stage Plan have been completed
- to provide information to the Project Board to assess the continuing viability of the project
- obtain authorisation to start the next state
- record 'lesson-learned' information for later stages
- to update the relevant Project documents

- to provide a *Stage End Report* and perhaps to
- set-up an *Exception Plan*.

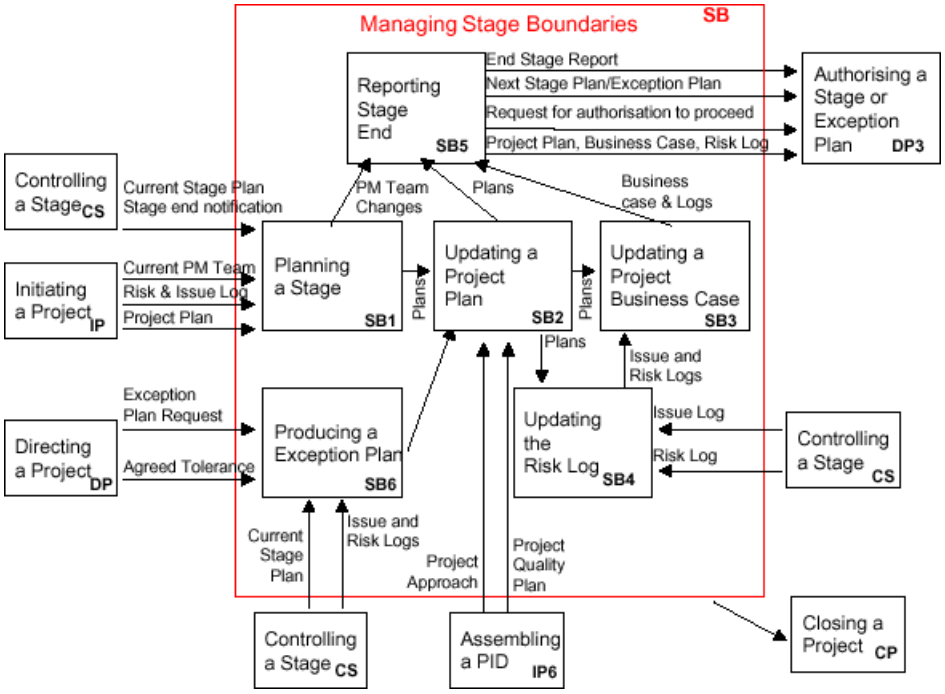

Figure 7.19: Managing Stage Boundaries [31]

7.3.8 Closing a Project (CP)

Once the Project is finally realised, it has be gracefully closed. Closing a Project is the respective process within PRINCE2 (figure [7.20]):

7.3.9 Planning (PL)

Planning is a common (sub-)process required by

SU6
IP2
SB1
SB6

- Planning an Initiation Stage
- Planning a Project
- Planning a Stage
- Producing an Exception Plan

Any plan has to include the following steps:

1. Establishing what products are needed
2. Describe products according and assign quality requirements
3. Determining the sequence order for products and their dependencies

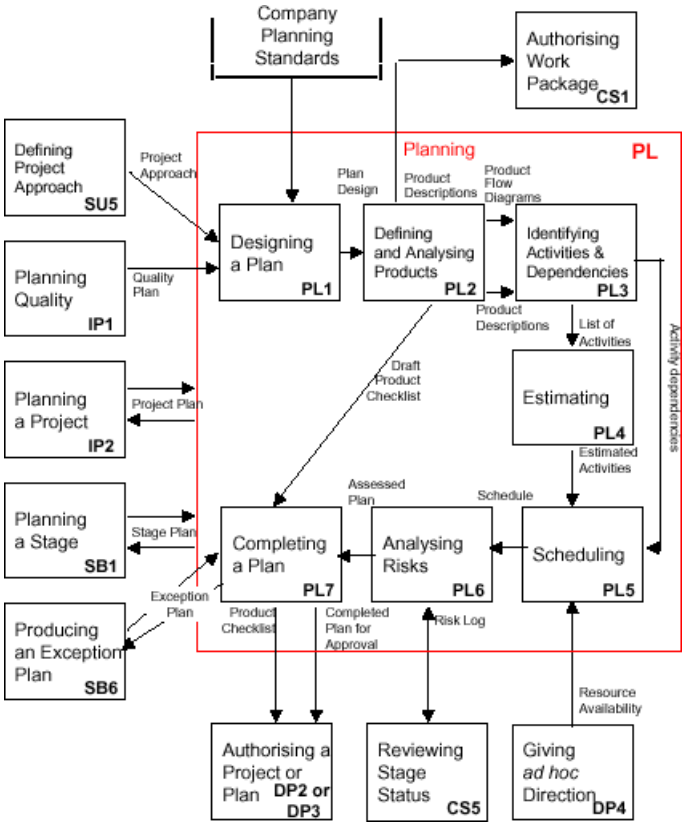

Figure 7.20: Closing A Project [31]

4. Check, when activities should be done and from whom
5. Estimate the amount of effort for each activity and the duration
6. Agreeing on Quality Control and the required resources
7. Calculate overall costs and efforts and make a budget forecast
8. Assessing risks
9. Identify management control points

Planning can subdivided into the following subprocesses (figure 57):

- Designing a Plan PL1
- Defining and Analysing Products PL2
- Identifying Activities and Dependencies PL3
- Estimating PL4
- Scheduling PL5
- Analysing Risks PL6
- Completing a Plan PL7

7.4 PRINCE2 – 2009

In the late summer 2009 finally a new version of PRINCE2 (not PRINCE3!) will be brought out by the OGC. Scope of the new approach is to make PRINCE2 more flexible w.r.t. the requirements of the actual project. Within the new issue sub processes are not mandatory any more, additionally checklists are provided and case studies are included. One further approach is, to align PRINCE2 with the new ITILv3 standard.

However the basic ingredients of PRINCE2 are kept and become more explicit:

7 Principles:

1. Continued Business Justification
2. Learn by Experience
3. Define Roles
4. Manage by Stages
5. Manage by Exception
6. Focus on Products
7. Tailor to suite the Environment

7 Themes (aka Components):

1. Business Case
2. Organisation
3. Quality
4. Planes (Project, Phase, Team)
5. Changes
6. Risks
7. Progress

6 Processes (unchanged):

1. Starting a Project
2. Initiating a Project
3. Controlling a Stage
4. Managing Stage Boundaries
5. Directing a Project
6. Closing a Project

Chapter 8

Project Management Body of Knowledge

Since several years, the *Project Management Institute* (**PMI**) offers a *knowledge-based* Project Management method condensed in the Project Management Body of Knowledge (PMBoK). Certificates can be achieved and yielding a Project Management Professional (PM) . Here, I can only give a brief introduction into the approaches of the PMBoK. PMI

8.1 Project Management Knowledge Realms

The PMBoK itself defines a standard for Project Management, including the description of tools and methods like the *Work Breakdown Structure* (**WBS**), *Critical Path Analysis* (**CPA**), which have been commonly accepted to be effective for Project Management. However for a successful project, the PM Team has to have knowledge about five different *realms*, including the PMBoK itself (figure [8.1])

- Project Management know-how according to the PMBoK
  - Definition of the Project Life Cycle-
  - Five PM Process Groups.
  - The nine Project Management Disciplines.
- Knowledge about the Standards and Rules regarding the *subject of project* (a product or a service) and how it will be applied.
  - Supporting infrastructure, e.g. lawyers, organisation of production and delivery, marketing, logistics, Human Resources.
  - Technical know-how, regarding production methods, Software development, engineering standards for the particular subject.
  - Specific know how in case of business-to-government relationships.

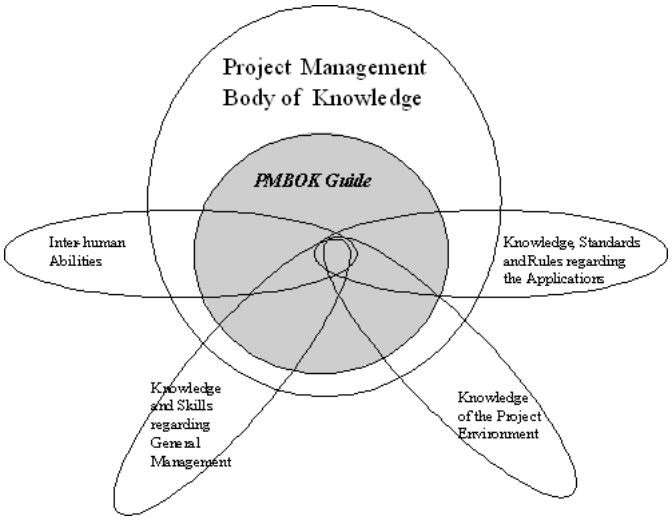

Figure 8.1: Project Management Knowledge Realms [21]

  - Know-how about the particular industry branch (automotive, aviation, pharmacy, financial services).

- Knowledge about the Project Environment.
  - The project's cultural and social environment and dependencies in order to build a successful project team.
  - International and political implications, eg. working hours, holidays, timezones.
  - Physical conditions for the project, office rooms, environmental considerations.
- Knowledge and Skills about General Project Management.
- Skills to Work and Communicate with people (inter-human abilities).

Groups

The PMBoK gives an outline of the five PM Process Group:

1. Initialisation
2. Planing
3. Execution
4. Control+Steering
5. Termination

Disciplines

Further, PMBoK introduces nine PM Disciplines

1. Integration Management,

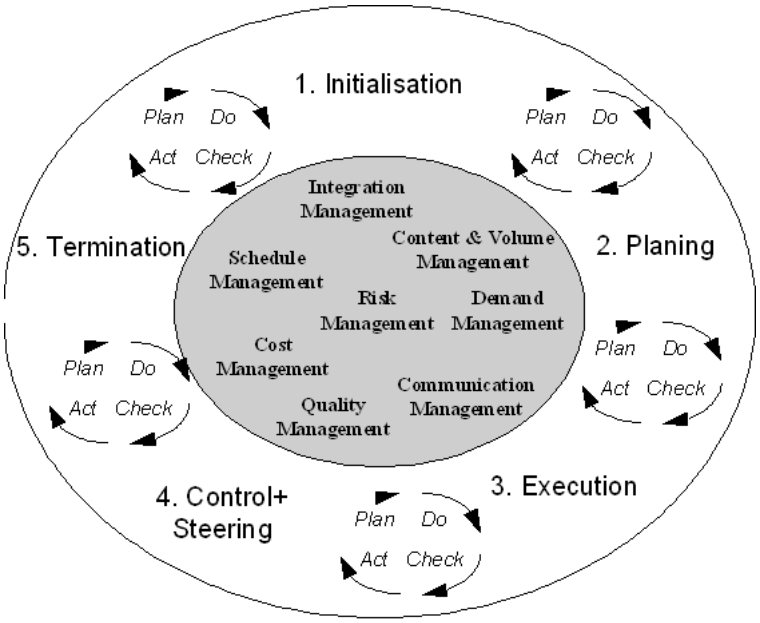

Figure 8.2: PMBoK Process Groups and PM Disciplines [21]

2. Scope Management,
3. Time Management,
4. Cost Management,
5. Quality Management,
6. Human Resource Management,
7. Communications Management,
8. Risk Management, and
9. Procurement Management

which will be discussed very condensed in the next sections. Figure 59 provides a bird's view of the process groups and PM disciplines. Like PRINCE2, the PMI defines a process (as part of a process group) to be dynamic and has to be optimised according to the *Deming Cycle* [8.3] (*Plan* $\rightarrow$ *Do* $\rightarrow$ *Check* $\rightarrow$ *Act*).

The finale result of the Project's subject is a *Product*, a *Delivery*, or perhaps a *Service*. Thus, the responsibility of the Project Management is to shape all processes required to derive to subject and to deliver all ingredients in time and in quality.

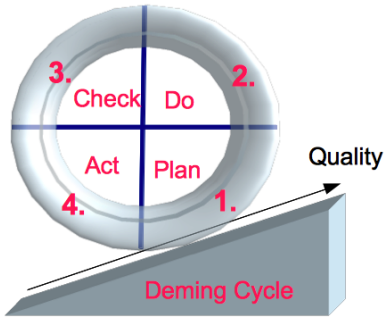

Figure 8.3: The *Deming Cycle*'s road to quality

8.2 Project and Product Life Cycle

PMBoK differentiates between the *Project Life Cycle* and the *Product Life Cycle*. The Project Life Cycle is broken down in phases, and the corresponding processes are grouped together in *Process Groups* (figure [8.4]).

Project Life Cycle

How the Project Life Cycle is structured, depends on the project's subject, of course but are typically organised sequentially. In PMBoK's terminology, *Milestones* are called *PM Output Values* and are pre-defined (figure [8.4]). Initially, the Project Manager has not only to determine the work packages (the tasks) for the project which is required for any resource planning, but rather has assign risks and costs too.

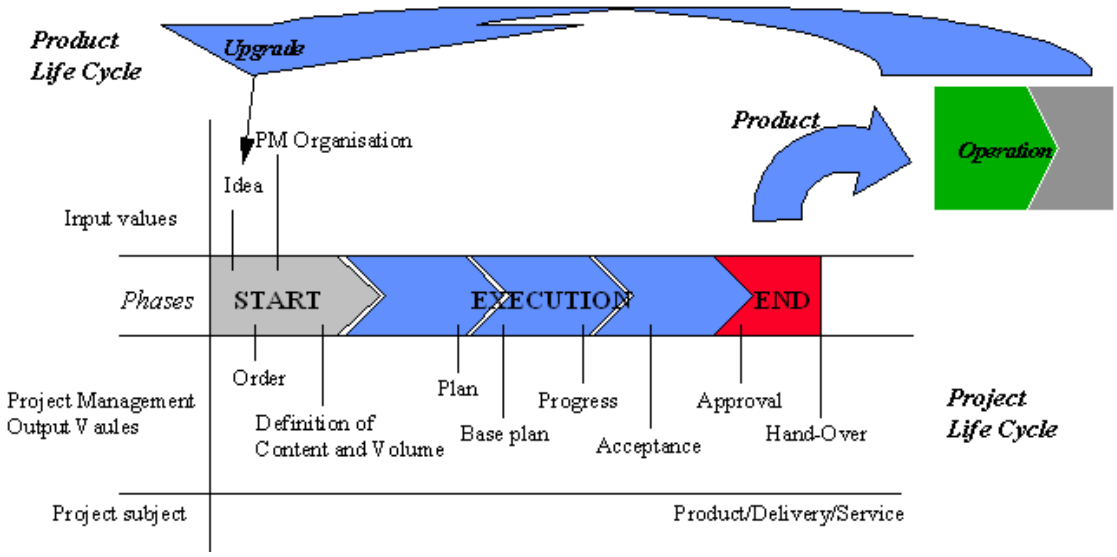

Figure 8.4: Project and PMBoK Process Groups and PM Disciplines [21]

8.3 Project Management Processes and Process Groups

According to the PMBoK, Project Management is "*applying knowledge, skills, tools and methods to fulfil the project's requirements.*" Project management is realised by a shaped and controlled processes, which uses *input values* and on return produce *output values*.

One particular output values are commonly referred to as *Milestones* (figure [8.4]), however PMBoK has a very generic approach to processes (figure [8.5]) and introduces specific diagrams and a *workflow* to couple the individual PM processes. For every PM process PMBoK defines line-by-line

Milestones

- a list of necessary Input Values and
- a set of expected Output Values.

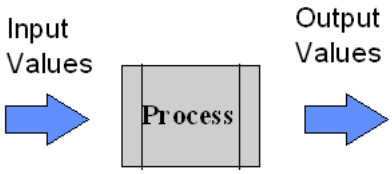

Figure 8.5: PMBoK's Process Diagram

The actual process is considered as event. The process is fed from the input values and generates finally the output values. Typically, processes are repetitious, thus they don't occur just once but rather more often. In this case, every process is subject of standardized process improvement, which commonly is known as '*Deming Cyce*' (figure 59). It is one essential task of PM to stoke this process improvement cycle and thus to achieve constant process improvements. Processes are grouped together by Process Groups (PG) which are shown in figure [8.6]:

Evens

Deming Cycle

We can consider

- Initialisation,
- Termination

as none-recurring Project Management Groups, where typically every process is only executed once, while within

- Planning and
- Execution

PG (sets of) processes are run often. Further, the Control and Steering PG can be understood as 'meta' Process Group (figure [8.7]).

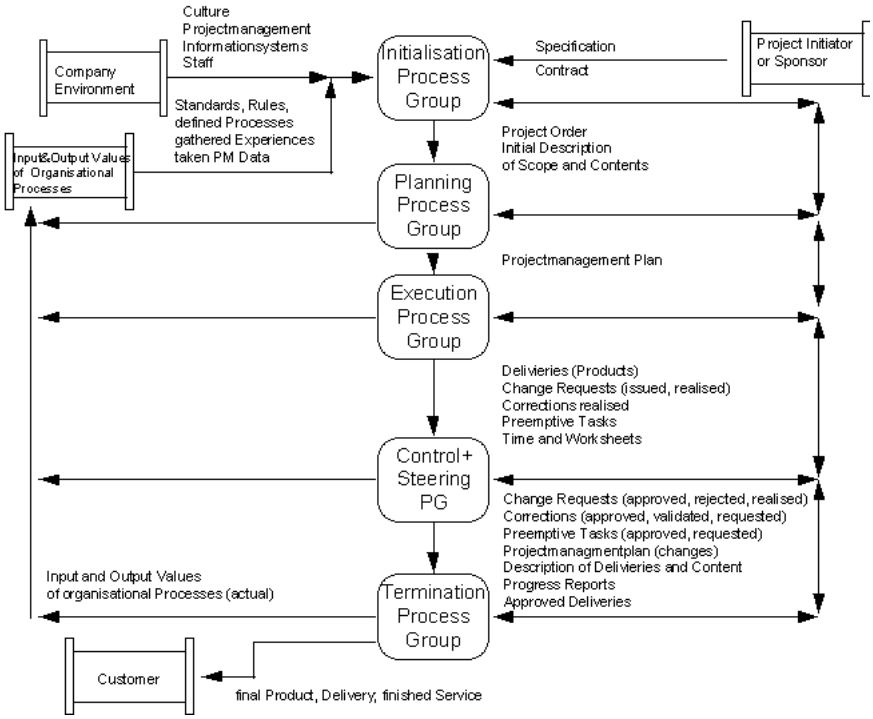

Figure 8.6: PMBoK's Process Groups and their interrelationships →

8.3.1 Initialisation

The processes during the PG Initialisation are limited, but rather important, as can be seen be the following subjects:

Development of the Mission statement for the project and how to achieve the project's approval. Defining the project's brief, providing information about content and volume of the project.

8.3.2 Planning

During the PG Planning, the following processes have to realised:

Development of a Project Management Plan

This defines, how the PM is set-up for the current project:

Project's Content and Volume

- Planning the *Project's Content and Volume* providing an estimate about the size of the project and the required resources.

Content Volume & Deliverables

- Definition of the *Content Volume* , on the other side, details the *deliverables*.

WBS

- Setting up the *Work Breakdown Structure* (WBS)

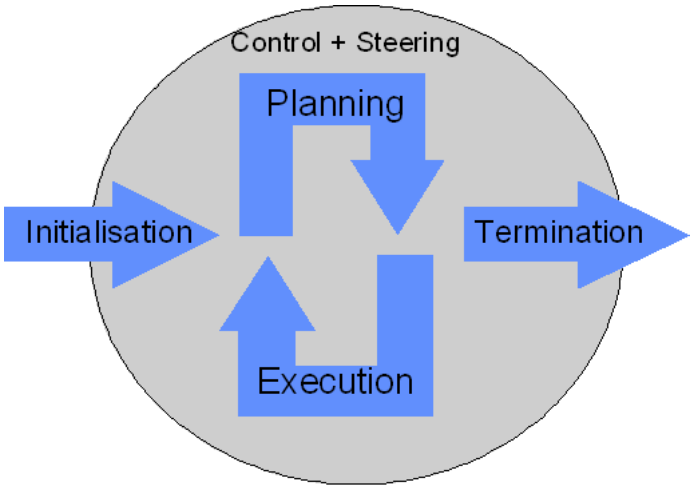

Figure 8.7: Project Management Groups and process shaping

- Splitting the deliverable into small sets of packages , in order to make them more manageable. — Packages
- Definition of *Tasks* builds up the logic to deliver and couple and individual packages. — Tasks
- Determination of *Task Orders* considering and describing the dependencies between the packages. — Task Orders
- Estimation of *Resources* of individual tasks and process steps. — Resources
- Estimation of *Duration* for the individual tasks. — Duration
- Determining the elements for the *Schedule* of the tasks including order, duration and required resources. — Schedule
- First breakdown and guess of *costs* individual steps. — Cost Planning
- *Cost planning*, while summing up the costs for logical groups and estimating to total project costs.
- Planning the *Quality* requirements for the project. — Quality
- Determining the *Staffing and Reporting* structures. — Staffing & Reporting
- Planning the *Communication* chains. — Communication
- Setting up a Risk planning . — Risk Planning
  - Identifying the risks of the project.
  - Provide a Risk analysis based on an probability and impact.
  - Numerical Risk analysis.
  - Planning the Risk management.
  - Planning Risk recovery activities/requirements.
- Planning *Suppliers and Ordering*. — Suppliers & Ordering
- Setting up *Contract Planning*. — Contract Planning

8.3.3 Project Execution

In the PG Execution, many recurring processes take place. In particular, we have the standard processes (figure [8.8]):

- Steering and Management of project Execution
- Realisation of Quality Management.
- Setting up Project Teams
- Continuously developing Project Teams
- Distribution of Information, in particular towards the project's Stakeholder.
- Requesting potential Suppliers and finally
- Chose the Suppliers and set-up contracts with those.

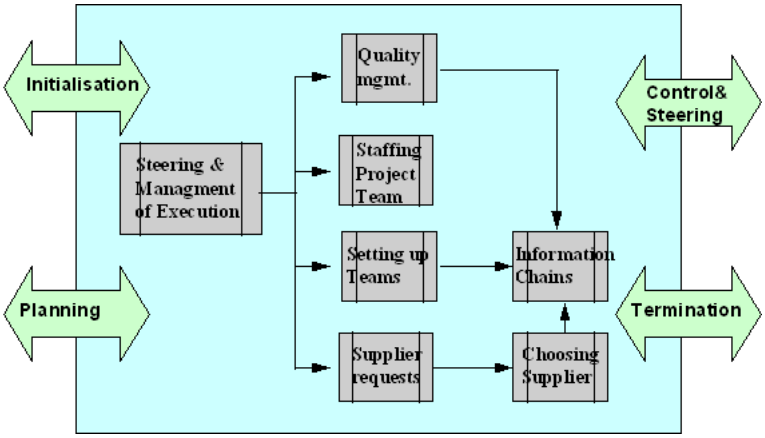

Figure 8.8: Processes in the Project Group Execution [21]

8.3.4 Control+Steering

Processes in the Control+Steering PG are set-up in order to watch and report activities in other processes, to identify potential problems and to allow corrective actions. In addition, all processes during the project's execution have to be compared against the original planning, and the plans have to be adjusted to the identified needs (figure [8.9]).

In particular the following processes need to be considered:

- *Supervision and Steering* of the project work including Time Sheets control, evaluating trends, improving processes and preparing status and progress reports.
- *Integrated Change Management* filling and approving Changes and validating their result.

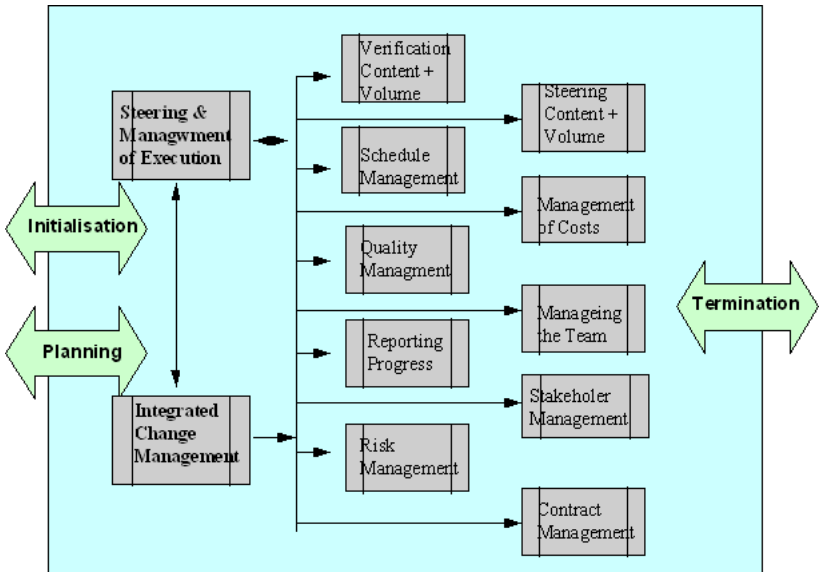

Figure 8.9: Elements of the Process Group Control+Steering [21]

- *Verification Content and Volume* allowing the final Acceptance of the project's deliveries.
- *Steering of Content and Volume* which may be required if changes in any of both aspects are necessary.
- *Management the Schedules and Time Tables*, thus any changes can be included in a coordinated manner.
- *Steering the Costs means* to spot additional costs and to adjust the budget planning.
- *Supervision of Quality Management* is to measure the achieved quality and to provide means to improve those and/or set up corrective actions.
- *Managing the Project Team* includes co-ordination, supervision and other tasks to improve the efficiency of the team.
- *Progress Reports* are required to identify the current state of the project, to provide an estimate of progress and perhaps a trend.
- *Stakeholder Management* is for informing the Stakeholders about the current project situation and to pick up their demands and perhaps including those into the project.

Stakeholder Management

8.3.5 Termination

During the Termination Process Group, the project will be firmly finished and all ongoing processes are terminated. Here, only two processes are relevant

- Termination of the Project including the final documentation of the project, which is required to

- Close all Pending Contracts due to the achieved approvals.

8.4 Project Management Disciplines: The know-how Groups

The PMBoK refers to sets of documents which are important for any project:

- *Project Mission Statement*: Includes the formal approval of the project.
- *Project Brief*: Including the project's content and volume and specific consideration of the tasks and the products/deliveries.
- *Project Management Plan*:
  Description of the individual steps of the project and how they will be realised and including:
- *Plan for Content and Volume Management.*
- *Schedule Management.*
- *Cost Management.*
- *Quality Management.*
- *Team Management.*
- *Communication Management.*
- *Risk Management.*
- *Order and Demand Management.*

8.4.1 Integration Management

The Integration Management is responsible to integrate the different elements of project management, which are located in different Process Management Groups (PG). Specifically it's tasks are:

- Developing the Project mission statement,
- developing the preliminary definition of project content and volume (Project Brief)
- development of the Project Management Plan,
- steering and co-ordination of the project execution,
- controlling and management of the individual project tasks while
- integrating Change Management and termination of the project.

The Integration Management is thus responsible to set up the projects management framework. In particular, here decisions have to be carried out, about the specific means and tools for Project Management, for instance a dedicated Project Management System. Often it is required to ask for external support, perhaps from inside the existing organisation or from outside by skilled and experienced consulters.

The main papers and proposals are the

- Preliminary definition of the *project's content and volume*, including Project Content and Volume
  - project and product goals in the context of the demand, standards, usage, and restrictions
  - specification of the resulting product or service and the required level of (formal) approval,
  - initial project organisation, risks, and estimated costs,
  - time-line and milestones with a first WBS,
  - requirements for Communication and Quality Management,
  - reporting chains and approval process.
- *Project Management Plan* including the already known depended Management Plans under consideration of PMP
  - the identified Project Management Processes and at which level they need to be implemented,
  - definition of required tools and methods for the processes,
  - identification of Process dependencies,
  - how tasks are executed and supervised,
  - communication needed, in particular to and from the Stakeholders,
  - the foreseen project life cycle and the adjacent phase planning,
  - reviews and reports for management in order to support decisions.

Another important outcome is a common understanding about the steering and managing the actual project execution among the project management team. The focus is to apply the same approaches (= how to do something) regarding the different tasks. In particular project management has to agree on

- approved correction methods to adjust the project results with the PMP
- approved preventive actions to reduce potential risks and negative impacts,
- approved error correction means, to eliminate mistakes/bugs identified during quality assurance.

The *Integration Management* has additionally to care about Controlling and Supervision of the ongoing project. Processes have to be defined (and later set up and realised) to Controlling & Supervision

- allow a comparison between the current realisation and the PMP,
- estimate of efficiency in order to predict potential corrections, or preventive actions not to run out-of schedule,
- analysis and determination of project risks, in order to make sure, risks are well identified and controlled,

- maintenance of an up-to-date project database supporting the creation of status and progress reports and allowing an calculation of further costs and schedule adjustments,
- supervising approved changes.

The last item is also part of the *integrated Change Management*, which is also part of the *Integration Management*.

8.4.2 Scope Management

The Scope Management (formerly called *Content and Volume Management*) touches the following issues:

- Collect requirements while defining and documenting the Stakeholder's needs regarding the project objectives.
- Define a detailed description of the project and the product.
  - Regarding the achieved Product (as a result of the project) it determines the characteristics and functions of the delivery and or service.
  - Regarding the current Project itself, it tells which steps have to be undertaken to realise the Product with the defined characteristics.

Figure [8.10] tries to visualise the relationship between Project and Product.

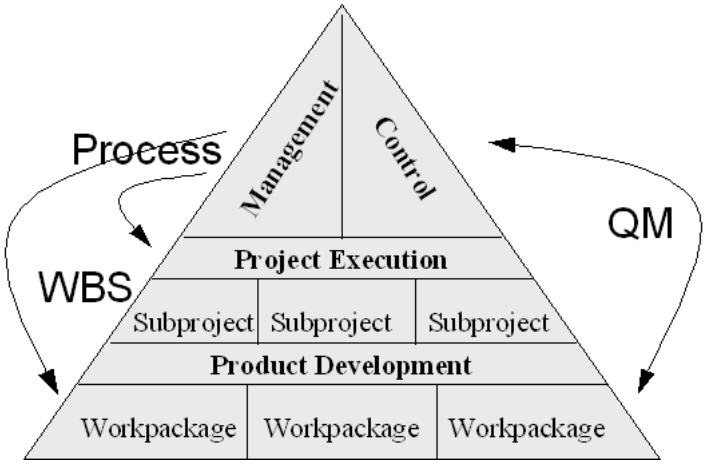

Figure 8.10: Relationship between Product and Project

While the product is statically broken into work packages (by means of the WBS) the actual realisation of a work package (under quality control) is a *Work Unit* under process control.

Work Unit

These two scopes yield the following results:
Scope 1: Verify and Control the Scope of the Project

- Project set-up: goals, requirements, and limits.
- Product definition: content and volume, final deliveries and acceptance criterions.
- Project execution: organisation, milestones, risks, costs, efforts and requirements, infrastructure, reporting chains.

Scope 2: Work Breakdown Structure

- Disassembling the product in *Work Packages* and assign *Work Units* to it. The work package defines the functional aspects (including interfaces) while the work unit is used to determine efforts, development duration, risks and costs. — Work Packages & Units
- The relationship between work packages might be complex and nonelinear. A WBS with the least interfaces (= dependencies) between the packages is believed to be (functionally) the best breakdown for the product. — Minimize Dependencies
- Typically, a hierarchical breakdown can be achieved, and each work package should be identified regarding it's order and hierarchy level. — Order Packages
- Once the dependencies between the packages has been determined, a *Critical Path Algorithm* can be used to achieve an optimal resource planning. — Perform CPA
- As input for the WBS the following information is crucial: — WBS
  - Structure of the project organisation (teams), this is the *Organisational Breakdown Structure* . — OBS
  - List of components required to build the product.
  - *Risk Breakdown Structure* detailing the project risks according to identified category. — Risk Breakdown
  - *Resource Breakdown Structure* telling what resources have to be used at each step for the product. — Resource Breakdown

8.4.3 Time Management

Schedule Management determines when a particular step has to be started, how long it will take, when it has to be finished and what are the dependent predecessors and successors.

The actual time line can be arrived from the WBS. However, at that level, the Schedule includes the aggregated efforts and costs. The PMBoK does not require a certain method here, but typically for smaller projects, a (linear) Gantt representation is sufficient, while lager projects require a more complex representation in terms of the NetPlan technique, allowing the inclusion and representation of alternate paths.

As a result of the Schedule one achieves:

- To-do lists of activities necessary for each step.

- The dependencies of the different steps including the required transition activities.
- The milestones, whether mandatory or optional to be achieved at a certain time.
- A potential reschedule of activities, subject of Change Management.

Adjust Schedule

A main part of project management is to constantly adjust the Schedule with the current project progress and thus to determine the dates of the forthcoming milestones. The adjustments are not necessarily considered negative, but they allow us:

- To precisely determine duration, costs, efforts for the particular step.
- Additionally, they are the base to do a numeric calculation of these respective values. While the first 'assigned number' in the Schedule may be educated guesses, the first correction can be used to allow a much more precise determination for the relevant values.

By analogy, we can now use the derived numbers and do a projection even on still ongoing (similar) activities. Regarding the whole Schedule, we achieve in addition:

- The possibility to determine the efficiency of the team.

SV

- An estimate of the *Schedule Variance*, that means measuring the impact on individual Working Packages regarding duration, costs, and efforts.

SPI

- Evaluating an *Schedule Progress Indicator* as a function of time, allowing to assess the quality of our planning with respect to the achieved project progress.

8.4.4 Cost Management

Cost Management has the main tasks

- to estimate the gross costs for the (sub-)projects
- to provide the input for a detailed cost and budget planning,
- to define methods to allow cost control and a strategy in case of overspending.

Cost estimates can be achieved while correlating the amount of work with costs.

Sample: Development of Windows 2000:
Lets assume Microsoft Windows 2000 with 40 MLoC. One real good programmer writes 100 LoC/day and costs 500 $/day. To finish W2K requires 400 programmers in 4 years. Development costs: 200 mio $, project costs roughly 500 mio $ (including management and marketing efforts).

The PMI uses for the cost calculation the *Earned Value Method* . The EV couples Milestones and the completion of the Work Packages with the costs. The initial key data are the EV

- *Budget at Completion* , which determines the initially assigned costs of the project. BAC
- *Currently Estimated costs at Completion* . EAC
- *Currently Estimated residual costs to complete* . ETC

During the projects evolution, Project Management needs to control the following important individual financial key data:

- The *Budgeted Cost of Work Scheduled*, which is the *Planned Value*. BCWS & PV
- The *Budgeted Cost of Work Performed* , which is the *Earned Value*. BCWP & EV
- The *Actual Cost of Work Performed* and thus the *Actual Costs*. ACWP & AC

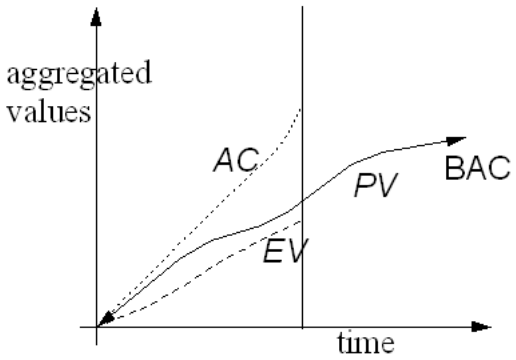

Figure 8.11: Time line of a project progress report in terms of cost development [21]

The project managers task is to balance the *Planned Value* (PV) and the realised *Earned Value* (EV) for every Work Unit with the Actual Costs (AC) for that component (figure [8.11]). To estimate the *Estimated Costs* to complete ETC and the *Estimated Costs at Completion* (EAC) three methods are commonly used:

Those estimates are typically used to provide Project Management a mean, whether the project is still in budget. A qualified PM software is able to determine those numbers in case the AC are correctly assigned to the work packages. A good knowledge of the project's costs w.r.t. to it's progress SPI will not only show, whether the project is well understood in terms of content and volume but also provide the required input values to potentially allocate additional budgets needed and serve as a qualified discussion base with stakeholders and sponsors.

8.4.5 Quality Management

The quality approach of the PMBoK follows the definition of the **ISO**. According to the A*merican Society of Quality* (2000), quality is seen as "*how far a group of intrinsic characteristics fulfils the requirements*". Quality management, on the other side, is always determined by the expectations of the Stakeholders.

The term "intrinsic quality" tells already, that quality is not necessarily related to functionality. A product might be simple from the functional point of view, but achieve a high quality standard. On the other side, complex products exist, which fail to achieve high quality. A good example is a simple text editor compared to complex word processing software.

For industrial processes, the PMBoK uses a differentiation between *precision* and *predictability*. Typical mechanical products for instances, are never 100 % precise, however their mechanical parameters (like length, diameter, strength) will vary due to production circumstances. A good production chain is able to predict the achieved values on a statistical base and provide a uniform distribution of those parameter according to known distributions (Gauss'ian or Normal distribution, Poisson distribution and others). In order to improve the achieved quality parameters (on a process related or selection base) several quality 'programs' are know:

- Total Quality Management (TQM),
- Six Sigma,
- Voice of Customer,
- Cost of Quality (CoQ) and others.

The PMBoK identifies the following important inputs for quality management:

- *Planning the quality*:
  Any planning has be streamlined with the initial expectations of the product. According to this, the quality standards of the product have to be defined and need to be balanced with efforts and costs. Further, here the (quality) transition criterion's are provided which are required to achieve a certain milestone.
- *QM methods and tools*:
  This depends of course on the product itself. Industrial production requires us of statistical methods, whereas software development often uses simple 'defect' counters.
- *Realising QM*:
  This might be an organisational question, since often already a QM department exist, which is responsible for Quality Assignment QA. One important issue is, that progress is 'trackable'. This means, any changes in the product, subject for QA, needs to be documented and reported.

- *Managing quality*:
  For industrial products, a quality chart (run chart) has to be assigned/attached to the product. For software, currently no corresponding scheme exits.

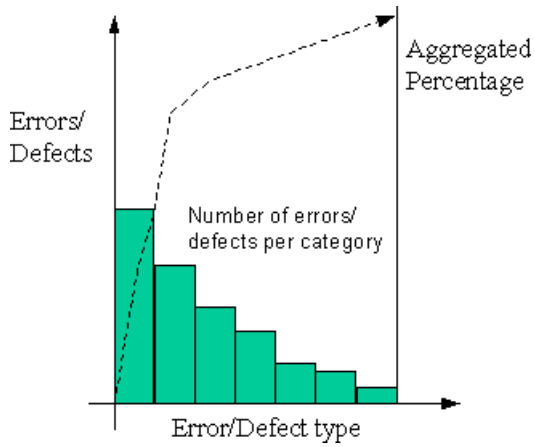

Figure 8.12: Pareto chart of error/defect distribution per category/module [21]

A common scheme for error assessment is to analyse the errors/defects and to sum them up per category or (software) module. A diagram showing such a distribution is known as *Pareto chart* (figure [8.12]). Pareto chart

8.4.6 Human Resource Management

Human Resource Management requires technical means to size and to plan and to control the team, and personal skills to lead and to manage the team members.

Among the technical means, the following means are useful (figure [8.13]):

- Templates: Descriptions of positions, rating sheets, and sheets for conflict management.
- Checklists: Description of rôles, competences, certificates of team members, security rules.
- Organigrams: Structural breakdown of the organisation in terms of rôles and competences (organisation chart).
- RACI matrix: Breakdown of rôles in terms of Responsibility, A countability, Consultancy and Informed of groups and individual persons regarding specific tasks.

Starting point for sizing the team is the WBS. After the PL has identified the required skills and qualifications for a task/rôle, the *Human Resource*

(**HR**) department will publish an advertisement to start recruiting. Depending on the need and urgency, the job advertisement is directed towards (1) internal staff members, (2) some dedicated tenders, or perhaps will be (3) open invitation to tender in a defined bid form.

After qualified tenders have been selected (based on their resumes), it is task to the PL or perhaps the sub PLs to interview the individual persons and finally to propose the chosen ones for contracting to the HR department.

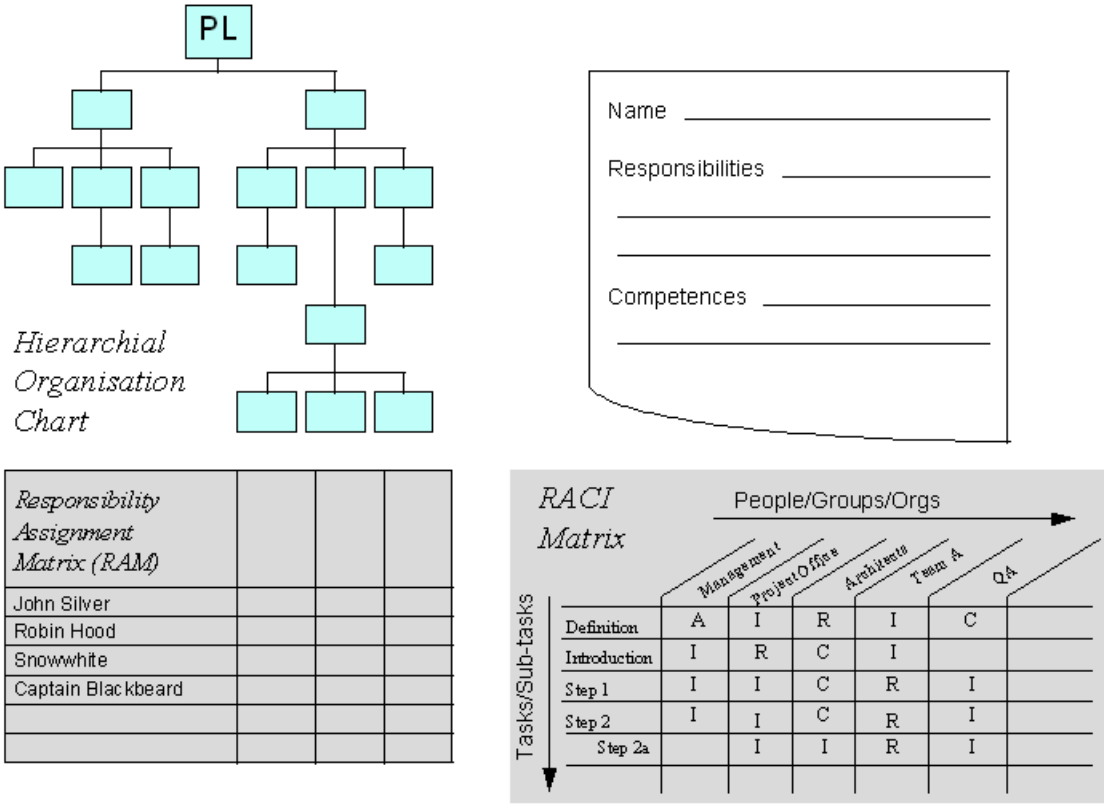

Figure 8.13: Tools for the PM to support Team management

Leading and controlling the team is subject of the personal skills of the PL. Main 'tools' are observation of and talks with the individual project members. Based on the achieved impressions (in comparison with the other group members), rating sheets can be a useful tool to monitor the persons and the group. For internal staff members it is common to finalise the rating sheet together the respective persons ("360^o feedback") . However, this might not be the task of the PL itself (as project supervisor), but rather for disciplinarian superior in terms of line manager. Improvement of teams means to improve the members individually and the inter-team relationships and behaviours.

360^o Feedback

Under the second category we can include:

Code of Conduct

- a 'code of conduct' for the team ('team spirit')
- defining the proper team positions for the individuals.
- organising team-building work shops,
- setting up the working conditions (office spaces) of the team such, an optimal inter-action is achieved. However, concerning today's communication possibilities there is a trend to construct 'virtual teams' which are coupled by Internet.
- Improving the team per individual may include
  - *additional qualification*: internally, externally, or based on CBT (Computer Based Training),
  - *bonus system*: overspending hours, finishing in schedule,
  - *mediation* and conflict solution strategies.

8.4.7 Communication Management

Known from computer communication, in order to achieve a revisable and reliable transmission of information, the recipient of the message has to confirm its reception.

Correct reception, however does not mean, that the message has been understood. It is task of the project management to use those means for communication which assures a maximum of understanding. This includes:

- The way of presenting a message by means of concise design and phrasing, proper gestural presentation and other visual means.
- The adoption of the message presentation according to the recipient. Thus the same message may be differently phrased whether it is directed towards the team members, the upper management or the Stakeholders.
- Whether the message is been transmitted over a 'formal' or 'informal' communication channel, and/or
- is due to a regular schedule, e.g. a progress or weekly report for which an 'incremental' message form could be suitable.

Further, the

- the subject and content, and the
- severity/priority of the message has a direct impact on it's presentation, as well as it is triggered by exceptional circumstances.

Progress Reports

For project management regarding the PMBoK approach, *Progress Reports* are indispensable. Those formal reports have to include information about

- the realisation status of a task or subtasks
- a performance measurement (see 10.4.4)
- the proposed termination date (see 10.4.3)
- QA status and means to improve quality (see 10.4.6)
- impact on the *Project Management Plan* PMP (see 10.4.1)
- approved changes,
- delivered products.

The reported values have to be included into the Schedules and into the *Planning Tools* (*Gantt charts*).

One important part of *Communication Management* is the *Stakeholder Management*. Apart from condensed progress reports, exceptional reports are import here. In order to achieve transparency in the current project realisation, these reports should describe the open items and challenges and how they have been solved or how solutions are proposed.

8.4.8 Risk Management

There is an ongoing discussion, whether Risk Management (RM) belongs to Project Management, or Project Management is a particular part of Risk Management. In the context of the PMBoK, the following issues belong to the Risk Management:

- *Risk Management Planning* – decide, where and for what tasks RM shall be considered.
- *Risk Identification* – determine and describe, which risks are important for the project's outcome.
- *Risk qualification* – prioritising risks according to impact and category, estimate probability for happening.
- *Risk quantification* – numerical risk analysis for each task and impact on the whole project.
- *Risk fighting strategies* – how to react on situation in which the 'risk' becomes apparent.
- *Risk management* – follow up identified risks, controlling open risks, and act on apparent risks according to the risk fighting strategies.

Risks are unavoidable for projects; if no particular risk exist, the task can be executed by line activities. Risk identification, qualification (in terms of impact) and quantification are the most important tasks of Project Management.

Risk Assessment
Risk Impact

According the WBS, it is necessary to develop a risk assessment in terms of a Risk Project Plan (RPP). This assessment could be broken down in terms of (1) milestones and (2) impact. The impact can be subdivided in (a) costs, (b) schedule, (c) quality, and (d) completeness.

How this assessment checklist is derived, depends on the contents of the project. Known approaches are:

- *Brainstorming*: The project team tries to identify the risk, maybe accompanied by a moderator. Brainstorming
- *Delphi-Method*: Particular Experts are involved to identify the risks anonymously. Delphi-Method
- *Questionnaire*: Team members, Stakeholders, and external Experts are asked for particular risks. Questionnaire
- *Determining the Reasons*: While the potential risks are grouped in categories, the risk assessment is refined and the risk sources are determined. Determing Reasons
- *SWOT Analysis*: Here, the *Strength*, *Weaknesses*, *Opportunities* and *Threats* are taken into account analyse the project under those conditions. SWOT Analysis

Those qualifying parameters have to accompanied by a quantitative risk analysis:

- *Impact*: Identifying the most important risks for the project in terms of completion. Impact
- *Expected Monetary Value*: Chances are treated as positive values, while risks are taken as negative values multiplies by their expectation values. The resulting sum is the EMV, which might not particular useful. EMV
- *Decision-Tree Analysis*: Here, two-dimensional calculations take place and allowing a diversification of results (for each decision chain). DTA
- *Modelling and Simulation*: The outcome of individual steps are estimated in terms of a Monte-Carlo simulation. Model & Simulate

As a result of an apparent risk, the Project Management has to act in terms of

(i) internal changes restructuring the teams
(ii) requiring additional support

which are both subject of Integration Management.

8.4.9 Procurement Management

Procurement Management within the PM deals with purchasing or acquiring particular parts of the project from an external party, whether in terms of products or services. Three major steps are involved:

1. Planning what to purchase/acquire.
2. Setting up the bidding (tendering) for the particular product/service and choosing a vendor.

3. Setting up and establishing the contracts with the chosen vendor and perhaps terminating it.

Certainly, the first step is to evaluate the 'Make-or-Buy' analysis. Here, PM has to investigate

- what are the costs of buying/leasing/renting/licensing + maintaining a product/software/service
- in contrast to building/developing/establishing the product/software/service as part of the project.

Often this turns out to be a political questions and answers are only determined by the costs but rather, whether this decision is aligned with the strategy of the company or the stakeholders.

In case contract with third-party companies (the chosen vendors) are settled, the commercial framework has to be mutually agreed upon:

CPF
- *Cost Plus Fee*
  The product is priced on the actual costs.

CPPC
- *Cost-Plus-Percentage of Cost*
  The product is price on the actual costs plus a cost-dependent fee.

CPFF
- *Cost-Plus-Fixed-Fee*
  The product is priced on the actual costs plus constant fee.

CPIF
- *Cost-Plus-Incentive-Fee*
  Parts of the expected costs (and fees) will be paid in advance, while the final price depends on quality and schedule.

T&M
- *Time And Material*
  Here, the product -as deliverable is not defined yet and costs will be covered based on activity confirmations.

	ETC	**EAC**
Sliding Estimates	From the efforts and spendings *Actual Costs* (current) **ACC** a prognosis for the cost developments is possible. Based on those number, a new *Estimate to Complete* **ETC** (again in terms of costs) can be facilitated.	Use the newly derived ETC and the current ACC to calculate the *Estimated Costs at Completion* EAC: $$EAC = ACC + ETC$$
Rough Estimates	Use the current Earned Value EVC as guess for the ETC: $$ETC = (BAC - EVC)$$	Consider the rest budget and the *Earned Value* (EV): $$EAC = ACC + BAC - EV$$
Corrected Estimates	The ratio between BCWS (= PV) and BCWP (= EV) is considered as *Cost Performance Index* CPI and evaluated for the individual milestones/work packages. The mean of the CPI is used as CPIC (current) to determine a corrected estimate: $$ETC = (BAC-EVC)/CPIC$$	Apply corrections as indicated by the CPI into the calculation: $$EAC = ACC+((BAC-EV)/CPIC)$$

Table 8.1: PMBoK cost calculation approaches

Chapter 9

Agility in Project Management

The pitch of the SW development starting the commercial use of the *Internet* about 1995 triggered a new evaluation of the SW development methods questioning the so-far relatively static methods.

It became apparent, that 'agile' and more 'iterative' methods are required, originally discussed as *Extreme Programming* and now known as *Agile Project Management* **APM**.

Many of those methods were tailored to cope with SW development only, however **Scrum** has an significant impact on general project management.

9.1 Extreme Programming

The door opener of *Agile Project Management* **APM** was certainly *Extreme Programming* **XP**: XP

- Founded by *Kent Beck*, *Ward Cunnugham*, and *Ron Jeffries*, in the year 2000, *Extreme Programming* **XP** became publicly known.
- **XP** is aiming to achieve a high-quality SW development.
- Here, the SW developer is constantly supported by a quality auditor, checking and auditing the code while it is written (and not perhaps later, in case bugs become obvious).
- SW development is realized in small teams, which are closely connected with the customer.

9.2 Feature Driven Development

The *Feature-Driven Development* **FDD** working model describes in the first place a rôle concept required for the SW development process. Here, the customer or the *principal* introduces the (SW) specification as sets of *features*.

FDD was initiated by *Jeff DeLuca* in terms of a five-tier process: FDD

1. Definition of the entire SW model
2. Setup a feature list
3. Plan a feature
4. Detail design of a feature
5. Develop a feature

9.3 Dynamic Systems Development Method

The *Dynamic Systems Develop Method* **DSDM** was initially defined until it's version 4 in 2007, as IT specific only.

DSDM

- While *quality control* was the main focus regarding the **XP** SW development. **DSDM** requires in particular the involvement of the *principal* as necessary third party.
- During the SW development process the *product* may be modified in the presence of the *principal* thus a rapid finalization possible.

DSDM recognizes three development phases and additionally five stages (*3 Phases & 5 Stages*):

1. Pre-project phase
2. Project phase including the stages:
   - Feasibility study
   - Business study
   - Functional model iteration
   - Design & Build iteration
   - Implementation
3. Post-project phase

9.4 The Agile Manifesto

The year 2001 was the 'official' starting point of the APM, since the most influential representatives of the agile SW development stuck together (*Kent Beck*, *Mike Beedle*, *Martin Flower*, *Kent Schwaber*, *Jeff Sutherland*, and others) and laid down the principals of agile methods in a *Code of conduct* known as *Agile Manifesto* [17]:

Agile Manifesto

> "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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

That is, while there is value in the items on the right, we value the items on the left more."

In addition, the *Agile Manifesto* defines *12 principals*:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity–the art of maximizing the amount of work not done–is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

9.5 Scrum

Scrum can be described as highly interactive SW development methods and consists of a limited number of elements and steps:

- The *Product Owner* provides a prioritized list (*wishlist*) of product attributes, which are known as the *Product Backlog*.
- Within a *Sprint planning* the *Team* picks up some items from that *wishlist* and discusses how realize those, resulting in a mutually agreed so-called *Sprint Backlog*.

Sprint

- A specific realization period – a *Sprint* – is provided to the *Team* to finish their work. The time period covers typically up to 4 weeks. Progress is reported and measured by means of daily *Scrum Meetings*.
- Main task of the *Scrum Masters* is to accompany the team's work and to remind and to monitor the realisation of the *Sprint Backlog* items.
- At the end of each *Sprints* a final realized and deliverable *work unit* shall be available, subject

  (i) to be presented to the *Stakeholder*,
  (ii) to be hand-over the customer – or perhaps –
  (iii) to be put aside (or scraped).

- The *Sprint* is terminate by means of a *Sprint Review* and *Lessons Learned*.
- The following *Sprint* cycle starts picking up new *work units* out of the *Product Backlog* and the team continues working (see figure [9.1]).

9.5.1 Scrum Rôles

Scrum as method deserves attention of the participant while acting accordingly to their foreseen *rôles* within the *Scrum* processes:

Product Owner

- The *Product Owner* is the representative of the customer and substitutes their interest within the project.
  The *Product Owner* is responsible to carry out the release planning, defining the product catalogue and prioritizing the individual items for the next Sprint (*Sprint Backlog*).

Sprint Team

- The *Team* is responsible for the *detail specification* of the product items and implementing (developing) the features while providing the required level of quality.
  Due to this requirement, the *team* includes members from different disciplines (SW developer, architects, tester) and having an average size 10 to 15 persons.
  The *Team* has sole responsibility for it's own organization and scheduling the development steps within the current *Sprint*.

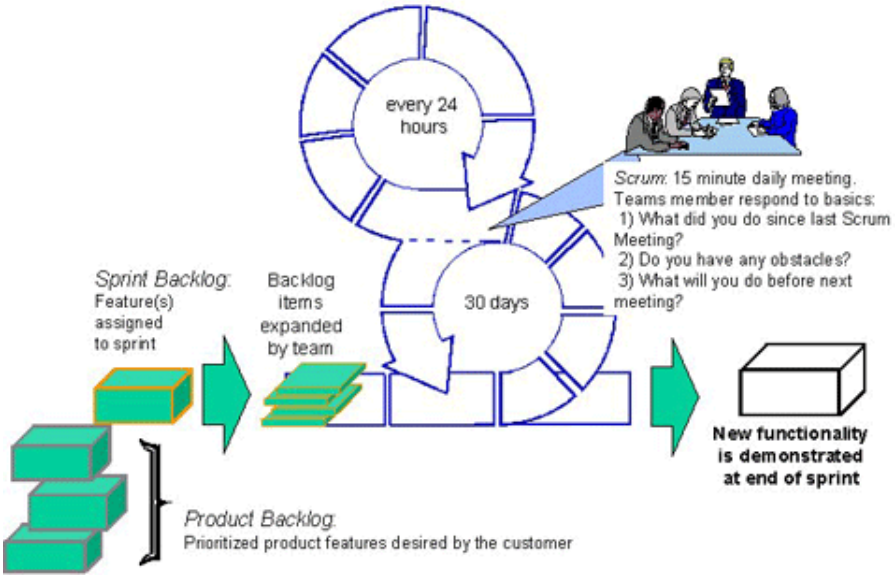

Figure 9.1: The Scrum development cycle [37]

- The *Scrum Master* initiates the *Sprint* and takes care about it's realization.
  Further, the *Scrum Master* is the de-facto coordinator of the team and *bridgehead* towards the *Product Owner* the *Stakeholders* respectively. — Scrum Master

9.5.2 Product Backlog

Starting point of the *project planning* is the *Product Vision*:

- The *Product Vision* includes the attributes and requirements of the product, which are in particular defined in the *Product Backlog* being the responsibility of the *Product Owner*. — Product Vision
- Scrum distinguishes between *Functional Requirements* **FR** and *None-Functional Requirements* **NFR**, which commonly define the explicit *quality requirements*.
- The elements of the Product Backlog are called *User Stories* and describe their functional and/or usage requirements. — User Story
- The *Release Plan* lays down the realization of the *Product Backlog* and provides it's logical order. — Release Plan
- In addition, an assessment of the required *resources* and the (expected) *development pace* of the teams; the realization time-frame respectively.

9.5.3 Scrum Artefacts

In general, *Scrum* distinguishes among *processes* (called *Events*), for instance a *Sprint* and *measurable items* described as *Artefacts* to be reported and managed by the *Scrum Master*:

- *Product Backlog*
  Pool of product attributes known as *User Stories* and items to be realized within the *release*. Responsibility is delegated to the *Product Owner*.
- *Sprint Backlog*
  The *Work Packages* assessed by the team for one *Sprint* in terms of individual *Work Units* also called *Tasks*.
- *Product Increment*
  The implemented/realized product attributes/items by the team during one *Sprint*.
- *Impediment Backlog*
  A list of (negative) conditions and (missing) tools acting as a handicap to realize a *Product Increment*. The *Scrum Master* is responsible to care about those.

9.5.4 Scrum versus legacy Project Management

Scrum provides the *team* a significant level of control regarding the actual project realization. Further, the *team* is allowed to contribute to the *detail specification* of the product:

1. Specification:
   The list of product attributes as defined by the *Product Owner*.

2. Design:
   The detailed *architecture* and *specification* of the product, commonly defined and agreed upon by the *Product Owner* and the *development team*.

3. Work Packages:
   The *team* decides upon the individual *Work packages*; not the *Product Owner*.

In opposite to the *Product Backlog*, the *Sprint Backlog* includes those tasks, which ought to be realized by the *team* within the current *Sprint*. While the *Product Backlog* is to be maintained by the *Product Owner*; the *team* is responsible to care about the *Sprint Backlog*.

9.6 The Sprint Process

In the Scrum spirit, development takes place in iterative cycles, the *Sprints*. Sprints

- At first, *Work Packages* are agreed upon in the *Sprint Planning* ('What ?').
- The *Team* is self-responsible for the implement ('How ?') the *Work Packages*.
- The duration of a *Sprints* shall not exceed 30 (working) days; it is actually this limit which determines the number of acceptable *User Stories* as result of the *Sprint Planning*.

As a result of a *Sprints* the team has realized a *Product Increment*, for instance, if a particular SW function has bee implemented, which was additionally quality-tested and well-documented. The final aim is – after a consecutive series of *Sprints* – to have the finale product ready for 'delivery'.

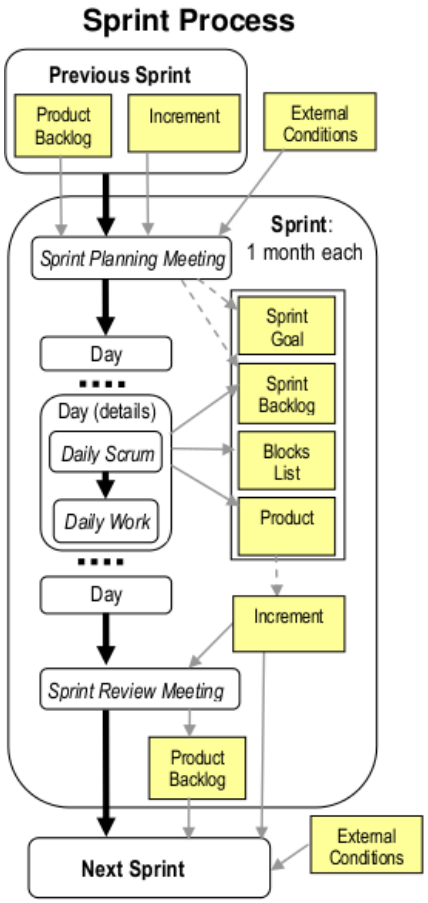

Figure 9.2: The Sprint process [9]

9.6.1 Sprint Planning

Starting from the *Product Backlog* within a one-day *Sprint Planning Meeting* it is discussed, which products/product items need to be realized within the next *Sprint*.

1. At first, the *Product Owner* details his ideas about the product items to be subject of the forth coming *Sprint*.
2. The *development team* discusses the resulting requirements and efforts accompanying the development of the product items.

Work Packages

3. Breaking down the *User Stories* into *Work Packages* the *team* decides, which tasks can be realized.

As a result of the *Sprint Planning Meeting* (lasting for typically 8 hours), the *team* is able to commit those objectives to the *Product Owner* and the *Scrum Master* which are identified as realizable *Work Packages*.

9.6.2 The Sprint Backlog

The result of the *Sprint Planning* is known as *Sprint Backlog* and includes

- a *description* of the single *Work Packages*,
- the *responsibilities* of the individual team members for each Work Package,
- the *development effort* attached to each Work Package, and
- the *time frame* required for the development.

9.6.3 Daily Scrum

Daily Scrum

Short term management of the *team* and measuring the results is facilitated by means of a *Daily Srucm Meeting*, supervised by the *Scrum Master* and lasting typically 15 minutes:

- Each team member needs to discuss it's achievement since the last *Daily Scrum*.
- New activities (due in the next days) are prioritized.
- Open problems and *Impediments* are discussed.

Impediments

Impediments resulting in the degradation of the daily work (e.g. missing SW, storage volume) need to be addressed to *Scrum Masters* who needs to record (*Impediment Backlog*) and of course to solve those.

Task Board

The central and public tools is the *Task Board*:

- As a result of the *Sprint Planning* a respective *Work package* is identified, treated as *task* recorded on one *line* and illustrated as 'moving' *Post-It* mentioning the *task* and the assigned team member. Tasks
- The *rows* however, show up the *state* of the *task*: Task State

  1. To-Do
  2. In-Progress
  3. To-Verify and
  4. Done

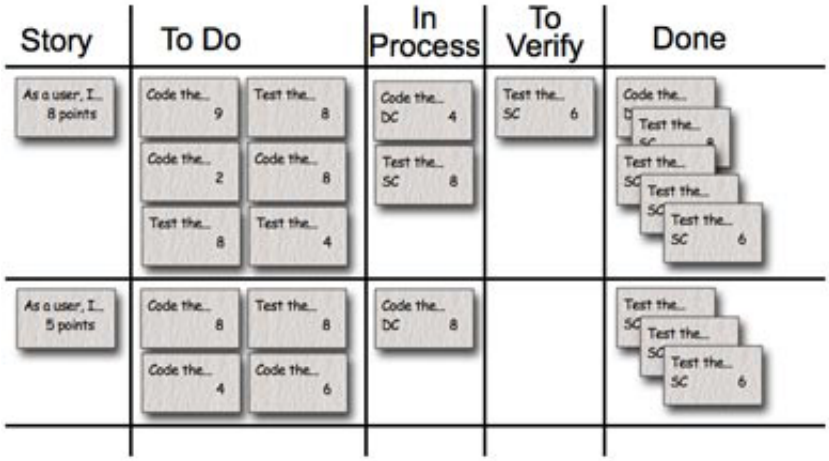

Figure 9.3: The Sprint Task Board [8]

9.6.4 Product Increment

At the end of a *Sprint* a measurable *Procuct Increment* shall be due and clearly visible on the *Sprint Board* in the state *Done*. However, often within projects, the team members have a different 'notion' what it means, a *Work Unit* or perhaps a *Product Feature* is really ⇒ **Done**! What means Done?

Due to the individual rôles in the project the understanding of 'Done' might be subject of mis-interpretation. Thus, it requires a common understanding which should be found while the team is initially set up:

- A feature is fully implemented, if all *functional* and *none-functional* requirements (**FR**/**NFR**) are realized.
- The product item as passed all functional tests (*Test Cases*).
- The *Usability* of the product item is verified.
- The product item did not exceed n defects of *Severity* 2 and no more severe ones.

9.6.5 Sprint Review

Sprint Review

Once the *Sprint* is terminated (after the defined period), a (public) *Sprint Review* takes place.

- The achieved results, the*Product Increments* are presented to the *Stakeholder*,
- which need to validate those against the (expressed) specifications and perhaps the (not explicitly expressed) requirements.
- The achieved results are the basis of the future *Sprint Planning* and perhaps
- modify the *Product Backlog*.

The *team* participate in the *Sprint Review* actively and discusses with the *Product Owner* and the *Scrum Master* the requirements for the next *Sprint*. *Sprint Reviews* are aiming to improve the *team's* qualification to perform the next *Sprints*.

9.6.6 Measuring the Sprint Progress

The *Scrum Master* takes responsibility to assess and measure the results achieved during the *Sprint* processes:

Figure 9.4: Sprint Burndown diagram [16]

Sprint Burndown Chart

- *Sprint Burndown Chart*:
  Starting from the *Sprint Backlog* the number of *Work Units* in state 'Done' are subtracted on a daily (or weekly) base from the number of

items in the initial *Backlog*. At the end of every *Sprint* the residual difference shall approach '0'.

- *Sprint Product Burndown Chart*:
  Instead of the *Work Units*, now the *Product Increment* is used and displayed as a function of time. This diagram shows equivalently the realized product items per *Sprint* period.

Sprint Product Burndown Chart

- *Release Burndown Chart*:
  If, however, the time-scale stretches the entire *release period* and not just a *Sprint* an estimate for the *Release* is possible.

Release Burndown Chart

- *Velocity Chart*:
  The *performance* of the *Sprint* teams can additionally be measured comparing (and displaying) the difference between the fore-casted *Work Units* in the *Sprint Planning meeting* against the final realized ones. Again here, the time-scale spans the entire *Release* cycle.

Velocity Chart

9.6.7 Sprint Retrospective

The final corner stone of every *Sprint* is the *Sprint Retrospective*. The achieved results of the last *Sprint* shall be assessed by means of

- the structure and 'composition' of the *team*,
- the actual *Sprint process* and management, and
- the available tools and resources.

Goal is to provide a quality assurance of the *Scrum process*, to support knowledge-transfer, and to address potential improvements. Those will be subject of the *Impediment Backlog* already for the next *Sprint* acting as *Lessons Learned*.

Chapter 10

Process Control with Six Sigma

Recurring steps within a project can be understood in terms of a *process*. The outcome of a process are measurable and are subject for improvements according to the **Deming Cycle Plan → Do → Check →Act**.

The 6σ process model assesses the following tasks:

- **Define, Measure, Analyse, Design, Verify**: DMADV
  Create a project to realise a product with an initial quality level of 4σ:
  - Define design goals and objects, which coincide the with the customer's expectations and company strategy.
  - Measure and determine the *Critical To Quality* **CTA** elements of the elements of *products* and the creating *processes* as well as the involved *risks*.
  - Analyse development and design alternatives based the *High-Level Design* and chose the best fitting design.
  - Design the required details and optimize those by means of a *verification study*.
  - Verify the resulting design, create prototypes, implement the production processes, and finally hand it over to the *process owner*.
- **Define, Analyse, Measure, Improve, Control**: DAMIC
  Continuously improve a production process:
  - Define the problems, perhaps based on the *Voice-of-the-Customer*, the project goals, and the companies strategy.
  - Measure the quality performance of a process.
  - Analyse the gathered measurement data and evaluate the *cause/-effect* correlation. Determine, whether this analyse includes all potential circumstances. Cause/effect correlation
  - Improve and optimise the processes.

– Control the achieved process statuses and continuously determining deficits yielding potential quality problems (*defects*) while setting up control systems.

10.1 Fishbone Diagrams

Comparable to a mind map a *Cause and Effect* diagram allows to understand dependencies and causes resulting in effects. *Kaoru Ishikawa* was the inventor of the so-called *Fishbone* diagrams.

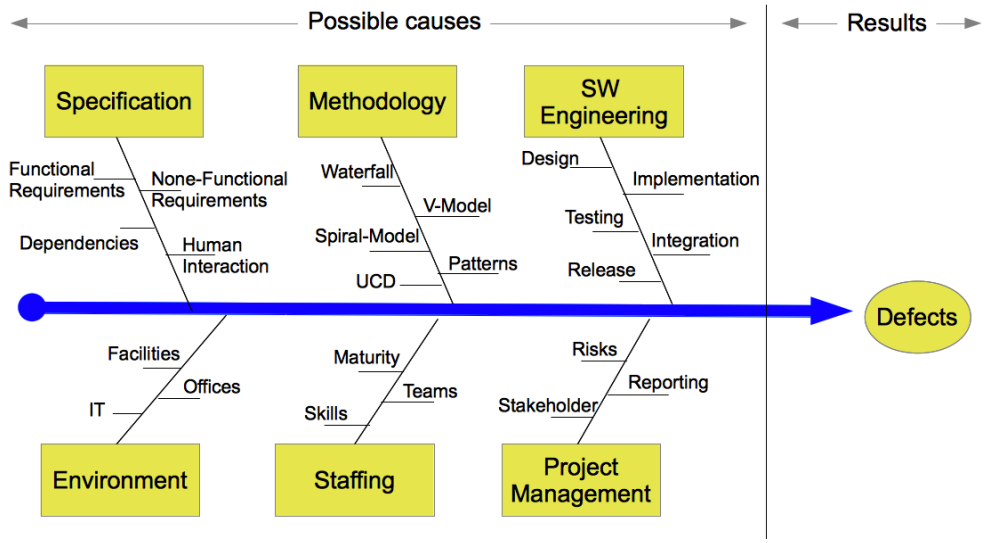

Figure 10.1: Fishbone diagram to asses IT SW development problems

The *Fishbone* diagram in [10.1] has been adopted to the IP project management case:

- The lower parts represents the common *project management* issues as known from the usual *Ishikawa* diagrams, where as
- the upper parts reflects the specifics of software engineering (*SE*) including the causes:
  - *specification*,
  - *methodology*, and
  - *engineering*.

10.2 FMEA

Identified defects and bugs can be systematically assessed by means of the *Failure Mode and Effects Analysis* **FMEA** (see figure [10.2]) method and may be analysed according to the following categories:

- Failure Mode Under which *circumstances* the error did occur.
- Failure Effect What are the *consequences* once the error had occured.
- Severity (ranging between 1 and 10) determining the *impact* of the error in relative to the usability of the product/step.
- Failure Cause The potential cause of the failure.
- Occurrence (again ranging between 1 and 10) providing a measure, what is the *frequency* of the failure to occur.
- Detection (on a scale from 1 to 10) assesses the likelihood with which the error is probably *detected*.
- Risk Priority Number `RPN = Severity` $\times$ `Occurrence` $\times$ `Detection`

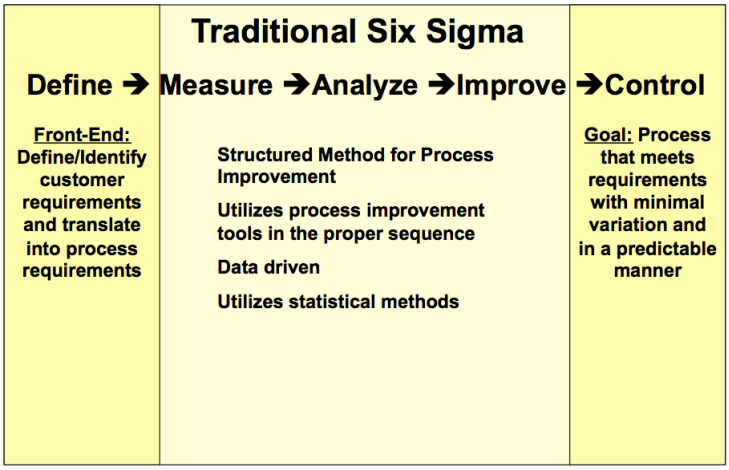

Figure 10.2: Setting up a FMEA analysis

Part VI

IT Product Development Management

While the discussed *Project Management* methods and frameworks can in principal be used for any kind of project, now we turn onto the specifics of IT (=SW) *Product Development Management* and it's merits.

Chapter 11

Software Development and Life Cycle Models

11.1 The SW Production Chain

Today it is recognised that any Computer software has it's own life cycle. This life cycle probably starts with the initial idea and is certainly not finished when the software becomes into operation. Since the first software projects have been realised (see figure 4), different life cycle models have been proposed and effectively used here. While our understanding of the software lifecycle is briefly shown in figure [11.1], we will discuss the heritage and the currently adopted models as well.

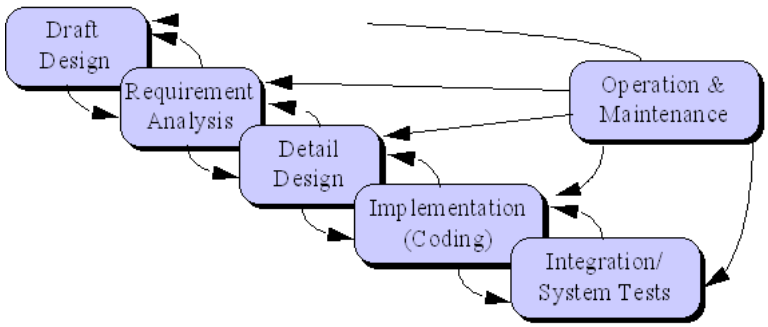

Figure 11.1: The main stages in the software life cycle [35]

In order to realize a high-quality software need to put attention on

- a careful analysis of the user's *requirements*,
- a fitting SW *design*,
- the subsequent *implementation*, and
- the required module/ system/acceptance-*tests*, and
- finally the *release*

regarding it's *procedural dependencies* and the *realizations conditions* – which are subject of the *IT project management* with respect to

1. *planning*,
2. *realization*, and
3. *supervision*,

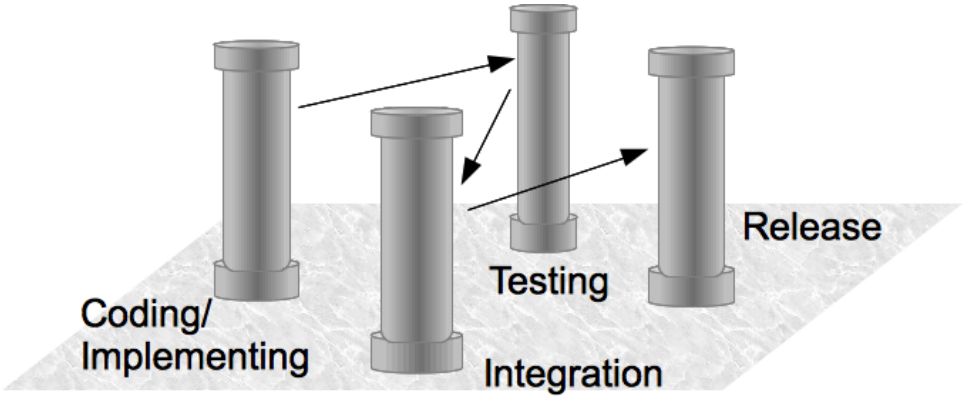

Figure 11.2: The four columns of the SW production chain

While the observance of the *functional requirements* **FR** defined within the *designs* and realized in the subsequent *implementation* and assured by the *module tests*, to achieve compliance with the *none-functional attributes* **NFR** the SW production chain is very important.

In particular with complex SW projects the *integration* of the individual modules to achieve running system can be considered of great importance.

11.2 The Waterfall Model

The Waterfall Model is probably the earliest approach to improve software development, as it has bee defined in 1956 already a part of the Semi-Automated Ground Environment (SAGE) as so-called stage-wise model. The final description of the Waterfall Model was carried out in 1970 and is shown in figure 81.

The main features of the Waterfall Model are:

- Recognition of feedback loops between the different stages and thus the impact of errors on one stage is restricted to the next stage and not passed to the final product.

- The inclusion of prototyping in the software lifecycle as a so-called 'build-twice' step.

As a result, the *Waterfall Model* (figure [11.3]) demands an existing verification/validation process before entering the next stage. Since 'open issues' are not accepted, in particular the development of interactive systems, requiring human intervention can not be handled efficiently – the concept of "Use Cases" was not present yet.

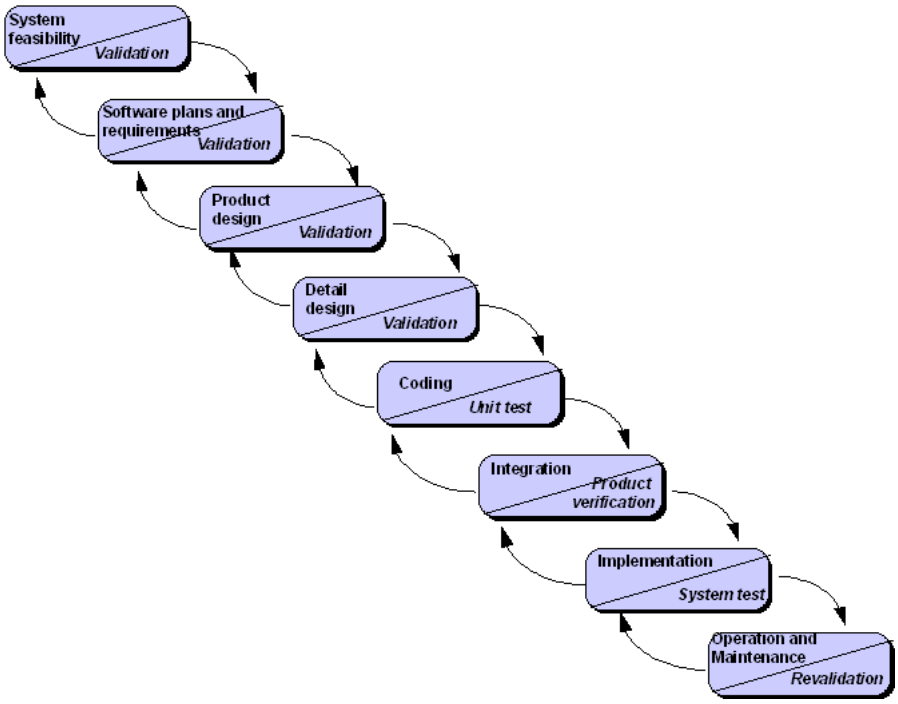

Figure 11.3: The Waterfall Model of the Software Live Cycle [6]

11.3 Use-Cases and Test-Cases

In today's software development, the design phase of the product or any component of the product includes a *Use Case*. The Use Case is a functional description of way, the product or the component is supposed to work. The event of *Object Oriented* programming has produced a certain schematic description of the components action flow, known as *Unified Model Language* **UML** which can be considered of a standardized description.

OO

UML

UML *Use Case* charts are the de-facto standard to represent dependencies as can be picked up from figure [11.4].

Use Case

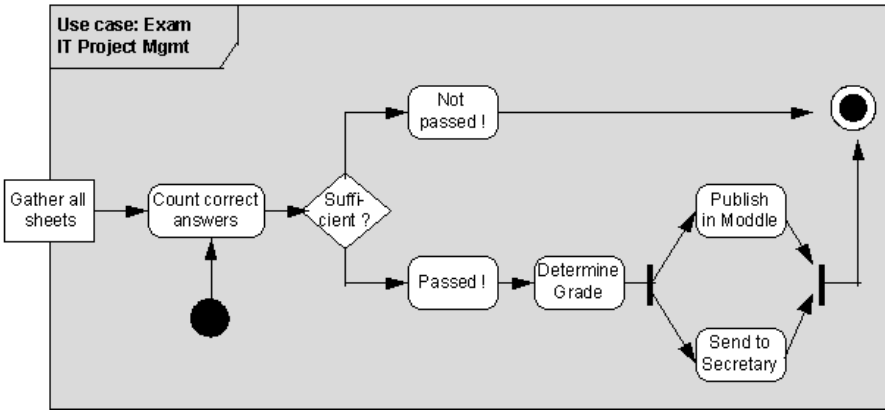

Figure 11.4: Use Case how to evaluate a written exam

The **UML** introduces several sets of charts:

General	**Mainly Programming**
Use Cases Charts	Class Charts
Packeting Charts	Object Charts
Usage- and Deployment Charts	Communication Charts
State Charts	Time Charts
Activity Charts	Interaction Charts
Component Charts	Sequence Charts

Table 11.1: UML Chart types [28]

Another common method is to specify the Use Case in an Activity Table while using a generic template [11.2].

Regarding software development each component, as described in terms of a Work Unit (PMBoK), has to be accompanied by a Use Case. Thus, the Use Case is part of the software design/development.

QP

According to the *Quality Plan* the QA department has to create *Test Case* suites. Here, the functional dependencies of the component are used to derive tests whether the software component is in conformance with the Use Case or not. Hence, any substantial tests have to prepared in term of Test Cases, which in turn depend on the existence of qualified Use Cases. In particular, to derive the necessary Use and Test Cases becomes difficult and

Test Cases

time-consuming for complex software products. As a result, complexity and quality of software products are believed to be opposing attributes.

As a result, only in the case of a well-documented software component, it's quality can be measured. A poorly documented software is almost impossible to gauge and effectively leads to a frustrated user (and tester !), or a user who does use only parts of the software's capability (while paid for the total). In spite of this, pretty often the QA department sets up tests based on (educated) guesses or needs to do a reverse engineering of the software component (however, typically inhibited by most companies policies).

A *Test Case,* should include the following (starting from the Use Case):

- Default behaviour: Provides the software component the expected results for default settings and input variables?
- Conditional behaviour: In case the component offers additional 'switches' / 'options' and/or 'arguments'/'parameters', do they work as expected (described)?
- Extreme behaviour: How does the component recognise input values out of specification and what are the results?
- Erratic behaviour: How does the component react, in case the required environmental conditions are not met/outside specification?

The qualified Test Case would detail the expected results based on a functional breakdown for the software component, while explicitly mentioning the conditions of the individual tests.

In conclusion, these dependencies very much underline the importance of a qualified design and development documentation. In particular, it is required to provide an exhaustive list of error and return codes for any software component. Since any documentation is expansive, in particular for development documents two approaches are common to ease this task for the developers:

- In-line documentation within the code: In the software code itself, relevant sections are documented with a specific mark-up language easy to parse and to collect.
  The programming language **PERL** has pioneered this, known as '*Plain Old Documentation*' **POD** [55].
- Documentation within the development framework: Some frameworks, like *Eclipse*, use context-sensitive information retrieval by means of a plug-in. Here, the current changes can be recognised and documented and in the same token saved (and retrieved) in a version-dependent manner.

11.4 The V-Model

Barry Boehm

The V-Model is a process-oriented approach for software development as part of IT project management and has been inspired by *Barry Boehm* and further developed in Germany, in particular for Government and here in addition for military projects. The 1986 published V-Model is now superseded with it's successor the V-Model XT.

Applying the V-Model for software developments can be seen as a cornerstone to comply with the ISO 9000 standards. It actually requires a particular infrastructure for software development and couples the responsibilities of *Project Management*, *Quality Management*, *Configuration Management*, and *Software Development* (figure 11.5).

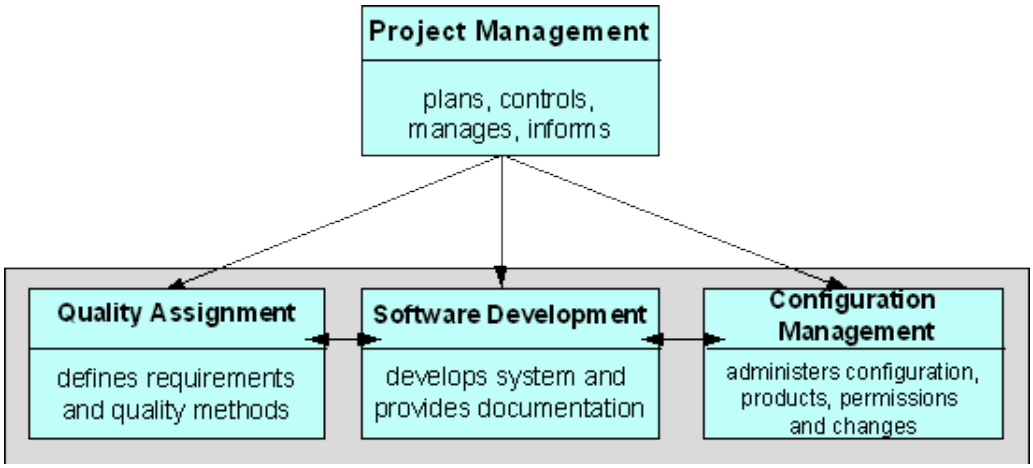

Figure 11.5: Interaction of the different PM levels according to the V-Model [35]

Unlike classical project management approaches which defines phases and transitions on a time line, the V-Model is procedure-driven:

- The V-Model defines a set of inter-related Events and Activities for planning and design, software development, testing, and the final roll-out process.
- In addition, the V-Model is document driven:
  During project execution, the activities are broken down per management level and the resulting state has to be documented to provide the required transparency for the project evolution.

Figure [11.6] depicts these dependencies and also provides a visual explanation for the term "V-Model (1997)".

The drawbacks of the V-Model approach are the following:

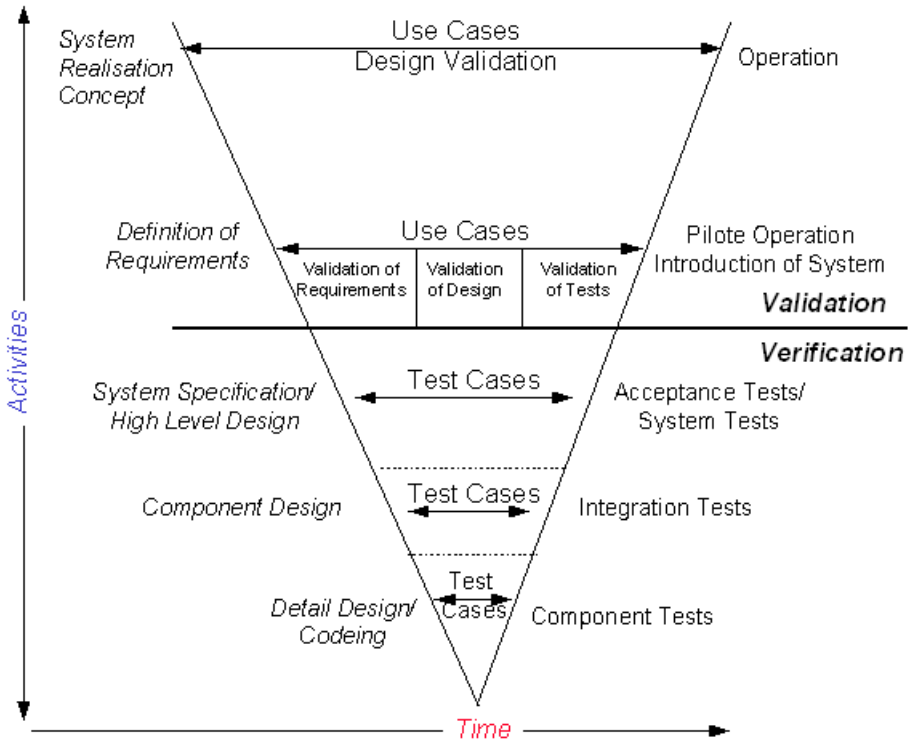

Figure 11.6: The V-Model

- It formalises the whole project management and software development (and demands the respective documents) while burdening it with a lot of administrative tasks.
- Software development is strictly pre-determined, which is unrealistic; and any change has to be approved.
- It provides too little interaction among the team members and thus does not exploit the team creative capacity.

11.5 The V-Model XT

Both the V-Model 1997 and the new V-Model XT is maintained by the German *BMI-KBSt* (*Koordinierungs- und Beratungsstelle der Bundesregierung*) and made public available (in a set of documents downloadable). Within the V-Model XT several new approaches are incorporated which are aligned with the achieved progress in the IT world: KBSt

- On the lowest level it introduces the Project Subject:
  - This might be a hardware solution,
  - a software system,

  - a complex system consisting of both,
  - an embedded system,
  - or perhaps system integration;
  - however it does not cover system services (which are out of scope).
- The definition of the Project Rôle tells who is using V-Model XT for co-ordination:

Customer & Vendor

  - the customer (Auftraggeber) managing one or several the vendors (Auftragnehmer),
  - the vendor (Auftragnehmer) employing it itself or other sub-vendors,
  - both customer and vendor using the V-Model XT.
- Thus we achieve the definition of the Project Type, whether it is

  - a customer-driven project,
  - a vendor-driven project,
  - or a shared project where both parties using the V-Model XT.

Thus, the V-Model XT adjusts explicitly to the typical situation for Government-related development, where the Government acts as 'client' and the 'vendor' is an external company; chosen after the bidding process.

The V-Model XT can be characterised as an incremental, adaptable, and model-driven procedure model. Unlike the original V-Model it does not employ formal documents, but rather *Computer Aided Software Engineering* tools are now considered for description and follow up. While the incremental structure of the V-Model XT is displayed in figure [11.7], it covers the main modules:

CASE

- PM: Project Management
- QA: Quality Assignment
- CM: Configuration Management
- PM: Problem- and Change Management

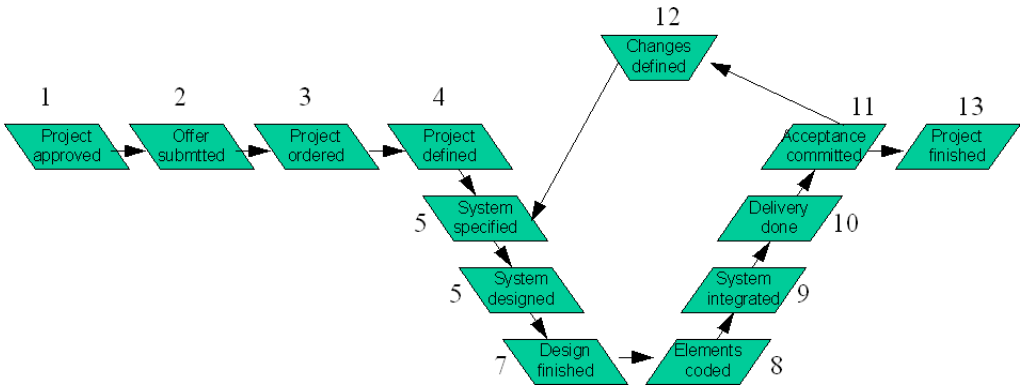

Figure 11.7: Incremental system development according to the V-Model XT (on the vendor side) [3]

Any change of the Product (or a sub-product) is called an *Activity* and has to obey the following phase changes (see figure [11.8]). Activity

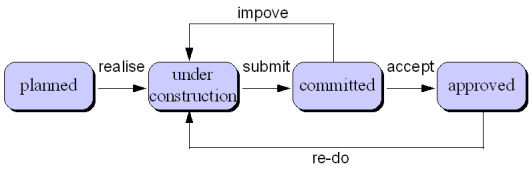

Figure 11.8: Phase changes of a product or sub-product

While the V-Model XT is mostly deployed in Germany it is recognised and used internationally, for instance for the *US American ITS* (Intelligent Transport System) and documented in the '*System Engineering Guidebook For ITS*' (figure [11.9]):

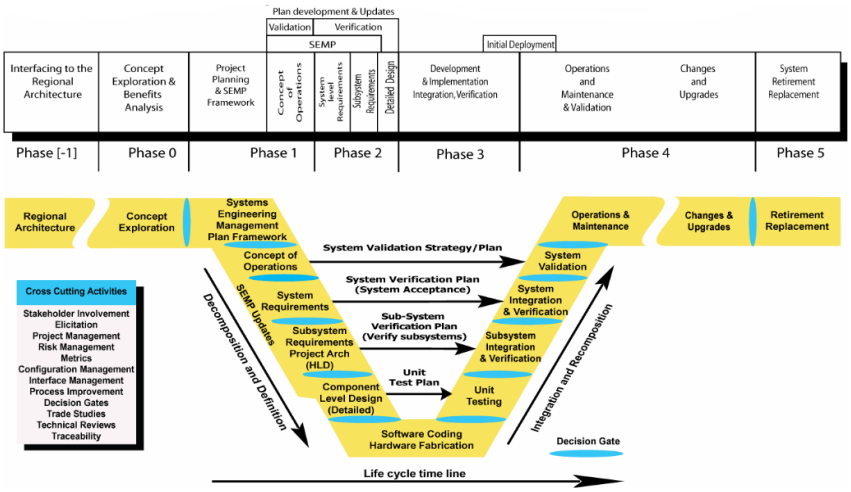

Figure 11.9: Adoption of the V-Model XT for the ITS [25]

11.6 The Spiral Model

The original Spiral Model was invented in 1988 by *Barry W. Boehm* and it projects the life-cycle of a (software) product on a (repetitious) spiral while defining four quadrants (Q1 $\mapsto$ Q4):

- Q1: Determines objectives, considers alternatives and constraints
- Q2: Evaluation of alternatives, identify and resolve risks
- Q3: Develop and verify, even consider next-level product
- Q4: Plan next phases

Unlike the other models, the Spiral Model is risk-driven; after each 'round' the risk to start with the next round is assessed. The risk assessment is based on certain key questions needed to be answered and documented:

1. What are the objectives?
2. What are the constraints?
3. What are the alternatives?
4. Which risks are involved?
5. How to resolve the risks?
6. What would be the result of the risk resolution?
7. What are the plans for the next phase?
8. What are the commitments?

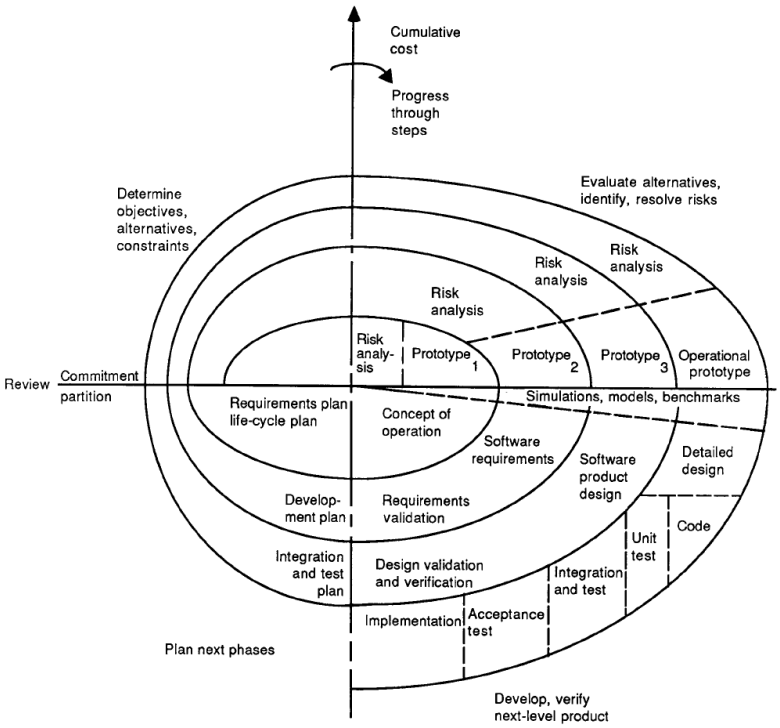

Figure 11.10: Spiral Model according the Boehm [6]

In this context, each cycle ('a round') can be considered as stage (figure [11.10]) :

- Round 0: Feasibility study
- Round 1: Concept of operation
- Round 2: Specification of the top-level requirements

Note: The Spiral Model introduced in chapter 7.2 (figure [5.14]) is a milestone-driven approach and thus differs in it's scope from Boehm's approach.

11.7 GQM

GQM is the acronym for *Goal/Question/Metric* and has been introduced by *Victory Basili* at the University of Maryland in 1983 and improved by *Dieter Rombach* 1988.

It can be seen as enhancements of the following :

- **QFD** *Quality Function Deployment* (developed by *Yoji Akao* 1966 at the Tamagawa University in Tokyo)
- **SQM** *Software Quality Metrics* (developed in 1980 by Marine for Metrics Incorporated) approaches and is suitable for Quality Management and Project Management as well.

GQM is a paradigm-driven approach; the result of any measurement is only valid in the context, as provided by the GQM plan. GQM Plan

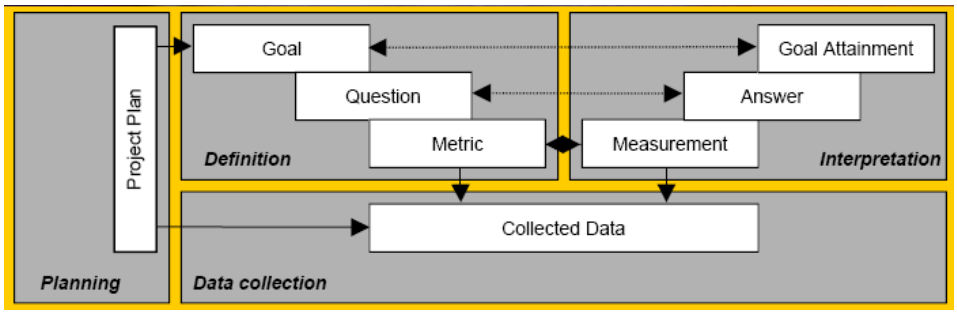

Figure 11.11: The four GQM phases [5]

Within the GQM approach, the following phases are considered (figure [11.11]):

1. The GQM plan
2. The definition of the questions and the metric for measurements
3. The collection of the measurement data
4. The interpretation of the measurements in the given context

Unlike other models, it respects the origin of the measurement data; sensible data are sheltered against random access.

Most import for software development is, that GQM provides a procedural language allowing a modelling of it's dependencies as shown figure [11.12]. For Eclipse, a plug-in FOCUS is available which allows the modelling.

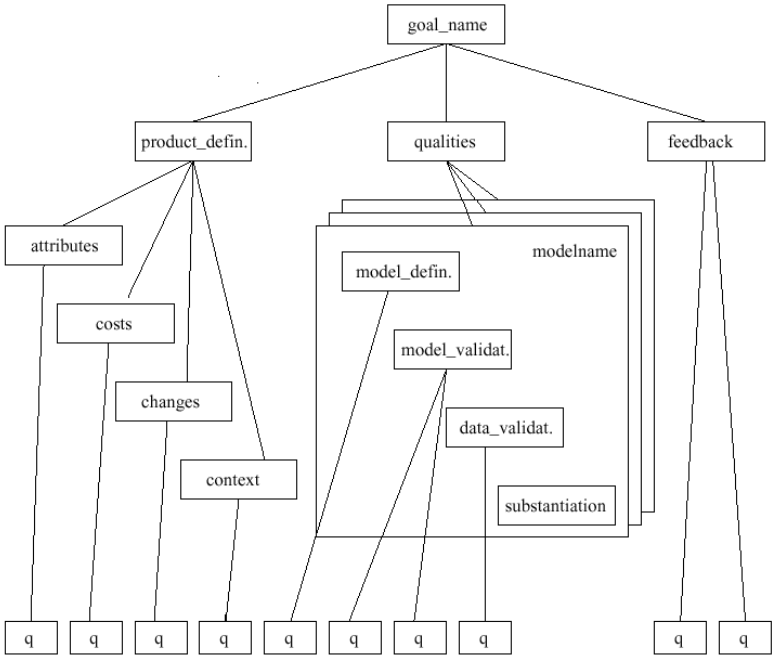

Figure 11.12: Schematic lay-out of a GQM plan [12]

11.8 The RUP Model

The *RUP Model* has been created by the company *Rational* (like *Rose*, *Clear Case* and other Rational software products) in 1996, which has been taken over by IBM. RUP is as procedure model for software development and currently available in version 9 (figure [11.13]).

OO

Like *Rose*, RUP uses an *Object Oriented approach* for software development and basically divides software development into two independent (orthogonal) streams:

- Disciplines:
  - Business Modelling
  - Requirements
  - Analysis & Design
  - Implementation
  - Test
  - Deployment – and additionally
  - Configuration & Change Management
  - Project Management
  - Environment
- Phases:

LCO

  - *Inception* (initial design providing the *Lifecycle Objectives* **LCO**)

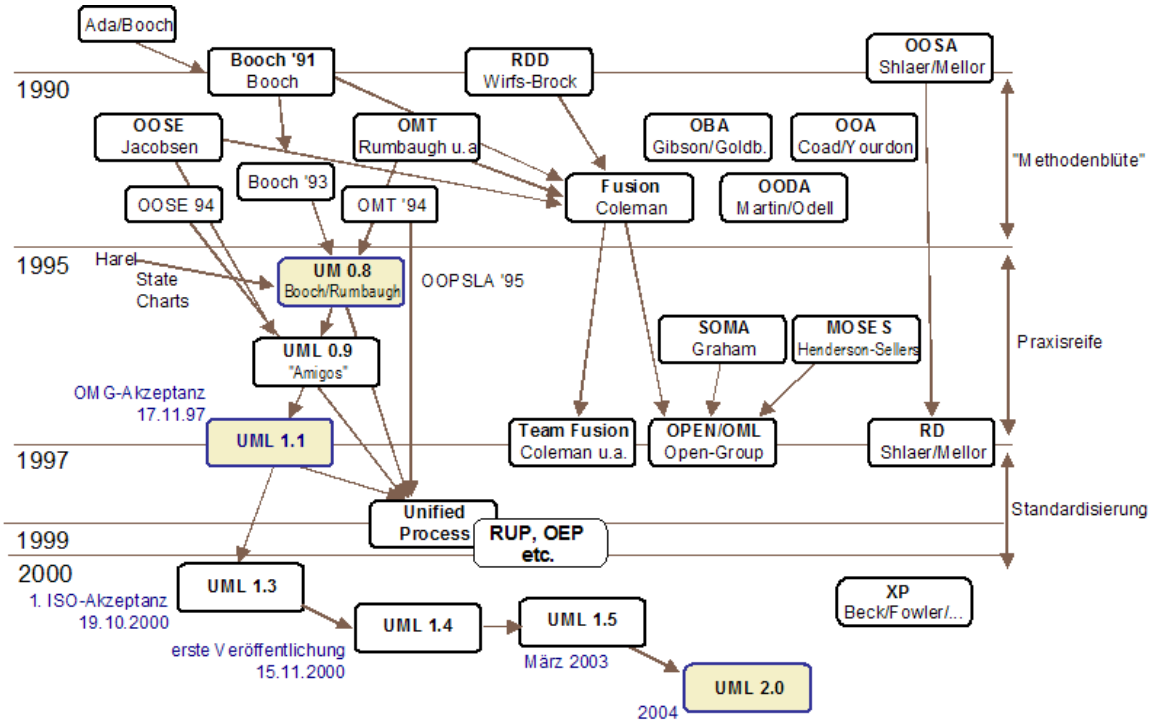

Figure 11.13: Development of OO modelling frameworks [52]

– *Elaboration* (design, yielding the *Lifecycle Architecture* Lifecycle Architecture) LCA

– *Construction* (coding, introducing the *Initial Operational Capabilities* **IOC**) IOC

– *Transition* (phase-change, responsible for *Product Release* **RP**) RP

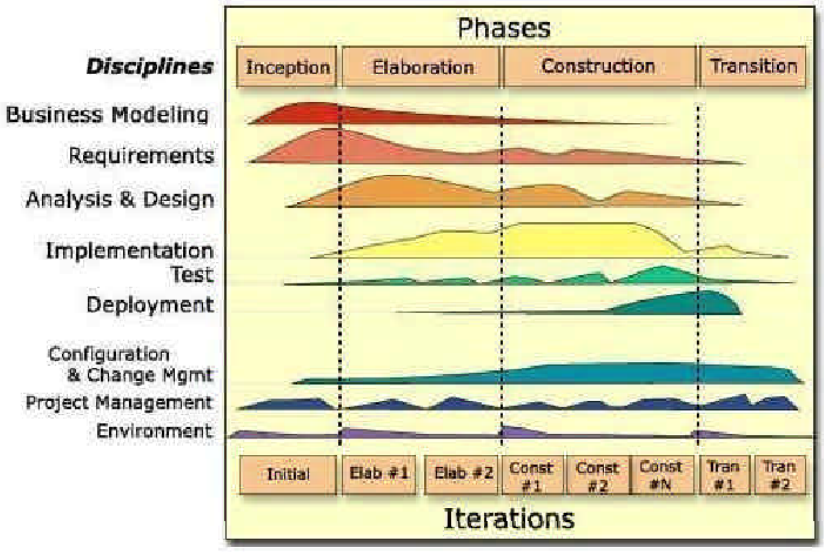

Figure 11.14: Intensity of the different RUP disciplines over the phases [39]

In this respect, RUP is an iterative-incremental approach to software modelling. Figure [11.14] demonstrates which disciplines are relevant to what extend in the different phases. For the different phases RUP defines sets of activities to be carried out.

11.9 User Centered Design: UCD

The *User/Human Centered Design* **UCD** (EN ISO 9241) expresses this in a more detail and defines the guidelines for *human interacting systems* (for example a Web page. `DIN EN ISO 9241-11` (*Requirements for the Usability*) provides three main criterion's – depending on the usage contexts:

Usability Requirements

Effectiveness

- *Effectiveness*: Precision and completeness, thus a user may realize a particular objective:
  May an average user realize the task by the system completely and correctly ?

Efficiency

- *Efficiency*: The effort required by the user to fulfill his objectives with the expected precision and completeness:
  May an average user realize the task with minimum efforts ?

Satisfaction

- *Satisfaction*: Absence of impediments and a positive stance to use the product:
  Is an average user satisfied with the system ?

The *usage context* depends here on the *user*, his *objectives*, the *tasks*, the available *tools*, and the *physical and social conditions*.

11.9.1 Usability Requirements for the User Centered Design

The standard `DIN EN ISO 9241-110` does not only provide quality *criterion's* but in addition details requirements for the *Usability*:

- Task adequate: An interactive system can be considered *task adequate*, if it supports the user to realize his tasks.
- Self descriptive: A dialogue is *self descriptive*, if the user at any time obviously knows where in the dialogue he is, and how to execute the current tasks.
- User Governance: The user is able to manage the interaction at any time.
- Predictability: A dialogue is *predictable* in case the usage context allows a recognition of the forthcoming actions and follows known paradigms and conventions.
- Fault tolerance: A dialogue is *fault tolerant* if despite of user input errors the purported aim can be achieved with minimal corrective actions.
- Customization: A dialogue is *customizable*, in case the user can modify the system-interaction and the display of information in order to achieve a personal adoption.
- Recognition support: A dialogue supports *recognition*, in case it guides the user to work with the interactive system.

11.9.2 Realizing User Centered Design

The realization of an **UCD** and in order to fulfil the quality and usability criterion's the standard `DIN EN ISO 9241-210` demands a particular *process* and *methodology*:

1. Analysis of the usage context
2. Definition of the requirements
3. Concept & design
4. Evaluation

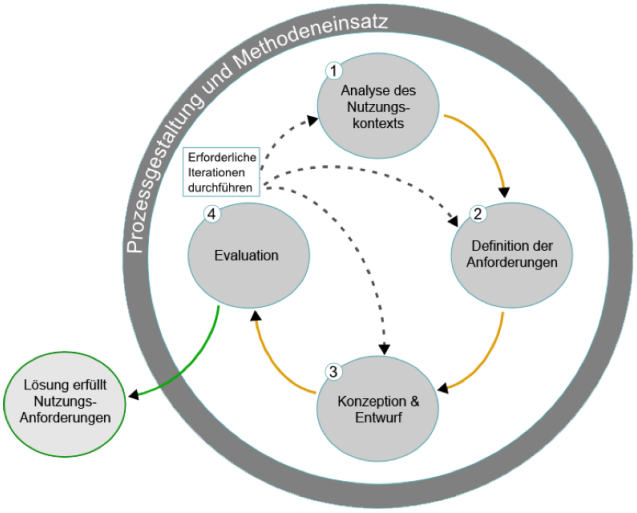

Figure 11.15: SW development requirements laid out by **UCD**

11.10 Software Metrics

Software Metrics is the approach, to define a measure for the given code (sizing). The measure is typically an one-dimensional value based on a certain method. The following methods are known:

LoC
- Number of Lines of Code (LoC), expressed as `kLoC` or `MLoC`.
- Number or words, characters, and files.

Function Points
- Number of functions or subroutines [51].

Classes
- Number of classes (for OO languages).

Which measure we use, depends on the programming language. The measure itself has to be unified, thus the same formula to determine the size has to be used. The rational behind that measure is to correlate it's value with the behaviouristic behaviour of software development:

- Law of continuous change:
  Once a Use Case is subject for a change, all relevant software modules have to be changed.
- Law of growing entropy: During it's evolution, the code becomes less and less maintainable.
- Law of a statistical smooth growth:
  A typical software project will produce a continuous and smooth growth of the code base.
- Pareto's law claims, that 20% of the code impacts 80% of the result.
- The law of the re-usability requires:
  "Don't solve any problem that has already been solved – check your

software repository for already existing solutions unlike develop them again."

- The Parkinson's law includes:
  "Work expands to fill the available time."
  "There's always time to do the right projects, but there's never time to do them over." "Adding manpower to a late software project makes it later."

- The Dr. Melvin Conway law predicts:
  "Organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations."

Table [11.3] provides a breakdown of the metrics for some E-Mail servers systems under UNIX. Here, we can see, that these different measures provide an estimate of the modularity of the systems.

Mail Transfer Agent	**Lines**	**Words**	**Characters**	**Files**
qmail-1.03	16.617	44.780	395.243	279
sendmail-8.9.1	55.059	179.376	1.229.121	54
zmailer-2.2e10	57.595	205.524	1.423.624	227
smail-3.2	62.331	246.140	1.701.112	151
exim-2.02	70.102	283.295	2.172.786	128

Table 11.3: Metrics for some UNIX Mail Transfer Agent

The most common approach for the software metric however, is to use the `LoC`. Figure [11.16] shows the dependency regarding the development of Windows NT.

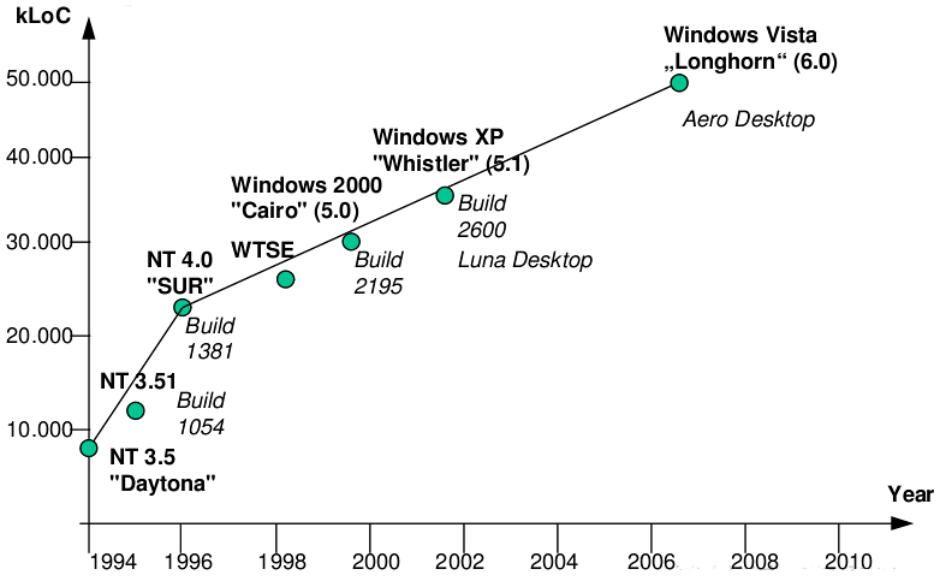

Figure 11.16: Code enhancement for the Windows NT product family

11.11 Software Modelling and CASE Tools

In software design several distinct design documents are relevant:

HLD

- *High Level Design document*, describing the products functionality in a bird's view, and the

LLD

- *Low Level Design document*, detailing the individual software components (and thus are also called *Detailed Design* documents).

In particular in Germany, the HLD is fed from the following input documents:

Pflichtenheft

- *Pflichtenheft* – the agreed specifications between the customer and the software vendor/developer.

Lastenheft

- *Lastenheft* – the requirement specification as provided by the customer.

CASE

The descriptions herein follow a formal modelling language and are provided in terms of charts. In order to produce the respective charts and to make them consistent for a given software layout. **CASE** tools (*Computer Aided Software Engineering*) are used in the design process. Today, three different modelling languages are common:

- **UML** - Unified Modelling Language
- **ERM** - Entity Relationship Model
- **SA/SD** - Structured Analysis/Structured Design

The CASE tools allow during the modelling process the determination of the dependencies of the software components. Thus, while deriving the WBS structure, CASE tools and be efficiently used to support this evaluation. Figure [11.17] shows a sample, how to model with the Eclipse plug-in *Together*.

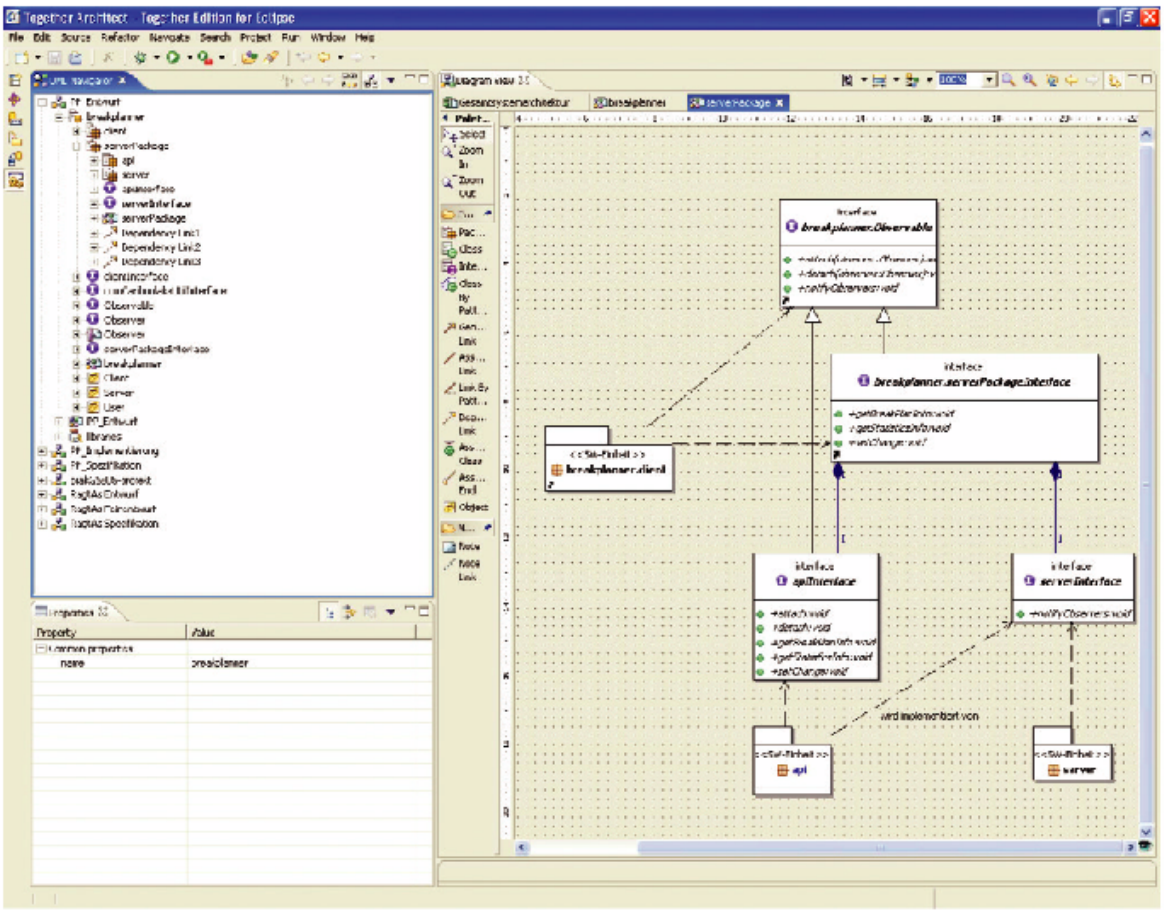

Figure 11.17: CASE modelling with Together [3]

Background, Exercises, Facts

The following Unix shell script provides for `c` and `c++` programs a measure for SW projects in terms of:

- Effective files – and herein –
- Used program words
- Program lines

Files can be nested in sub directories.

```
#!/bin/sh
NFILE=0
NWORD=0
NLINES=0

FILES=$(find . -name "*.c")
FILES="${FILES}$(find . -name "*.h")"
FILES="${FILES}$(find . -name "*.cpp")"

for FILE in $FILES
do
  NLINE=0
# WORDS=$(grep -v "^#" ${FILE} | cpp -fpreprocessed \
  WORDS=$(cat ${FILE} | cpp -fpreprocessed \
    | tr '\\' '/' \
    | sed 's/[_a-zA-Z0-9][_a-zA-Z0-9]*/x/g' \
    | tr -d '\012' | wc -c)
  NLINE=$(cat $FILE | grep -v ^/ | grep -v ^$ | grep -v ^\* \
    | grep -v ^# | wc -l)
  NLINES=$(($NLINES+$NLINE))
  NFILE=$(($NFILE+1))
  NWORD=$(($NWORD+$WORDS))
  echo "File $FILE uses $WORDS program words and $NLINE code lines."
done

echo "A total of $NFILE files examined with $NWORD program words \
      in $NLINES program code lines."

exit 0
```

<Name of the Use Case>	**<Version>**		**<Date>**
Author	*My Name*		
Brief Description			
Primary Actor			
Secondary Actor			
Trigger, Predecessor			
Event, Successor			
Technical dependencies			
Open items, Remarks			
Step	Actor	Activity	Junctions
1.			
1.1			
1.2			
2.			
Exception			
Variant			
Option			

Table 11.2: Template for an Activity Table of a Use Case

Chapter 12

Software Quality and Defect Management

Software projects (as part of IT projects) result in a piece of code which can be

- directly installed executed on a particular Operating System (OS) platform (like Windows, UNIX, MacOS) or can
- delivered in source code, requiring compilation and linking of the respective programs prior of execution.

Regarding quality, the software has to fulfil particular requirements:

1. *Usability*: The software as to fulfil the defined tasks and the published *use cases* (*'works as designed'*). Usability
2. *Conformance*: The software has to comply to the published functional aspects (*'works as expected'*). Conformance
3. *Absence of bugs*: The software should be reasonable bug-free (*'works under all circumstances'*). Bug free
4. *Security*: The software should neither directly nor indirectly impact the security context of the user (*'works without security impact'*), except where explicitly stated. Security
5. *Performance*: If performance is not part of the Usability, the software should use as little system resources as possible and achieve maximum performance (*'works with little system impact and high performance'*). Performance
6. *Maintainability*: The software is easy to upgrade (*'seamless upgrade possibility'*). Maintainability

These goals can only be achieved

- by means of a qualified software design, which fits to the respective tasks and considers the requirements,
- with a well-suited OS and perhaps necessary *middleware*,
- developing the software in a (quality) controlled environment with supporting infrastructure, and
- by aid of a qualified defect tracking and documentation system.

On the other hand, the software quality is independent of the programming language itself, and hence whether the code is actually executed via an interpreter (script, macro), as byte-code (within a virtual machine), or directly as binary (including OS loader statements). Of course, the choice of the programming language has a substantial impact on the performance (and occasional on security too). Figure [12.1] tries to outline the conflicting attributes in software developments.

The chain:

$$\boxed{\text{Design/Planning} \Longrightarrow \text{Coding/Development} \Longrightarrow \text{Control/Improvement}}$$

is typically limited by budget and time-to-market conditions. Shortages in any of those steps has a direct influx on the quality of the software product.

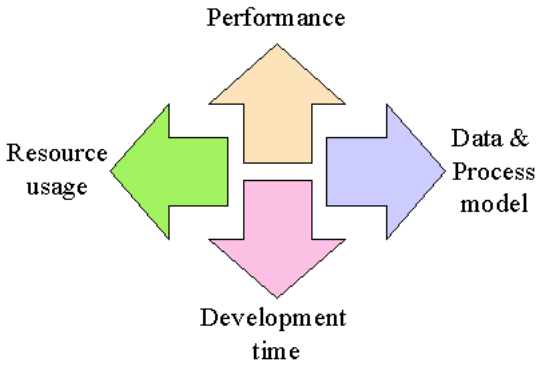

Figure 12.1: Conflicting attributes in software development

12.1 The Software test cycle

While testing software, we consider

1. *White Box* or *structural* tests.
   Those are typically facilitated be the developer itself, who is knowledgeable about his own code and the expected results.

2. *Black Box* or *behavioural* tests.
   Those are derived from the respective *Use Cases* and poured into *Test*

Cases to be executed by a dedicated test team as part of the *quality plan* **QA**.

3. *Beta tests* aka *live tests* at the customer site.
   Different from the guided *Black Box* tests those are unconditional or '*ad hoc*' and thus reflect the customer's behaviour and expectations.

The scope of the SW test is typically described by a *phase model* consisting of:

- *Unit* or *module* tests
  C programs (consisting of several functions) or a Java class is subject of a functional *White Box* test, to determine it's principal fitness.
- *Component* or *sub-system* tests,
  consisting of several depending modules to be commonly *White Box* or *Black Box* tested.
- *Integration* tests
  are indispensable for complex SW projects and determine the the mutual fitness of different components (and perhaps different vendors).
- *Acceptance* also called *User Acceptance* tests **UAT**
  are the foundation for the commercial delivery of the product.
- *Performance* and *regression* tests
  complement the *integration* tests with respect to the requirements met in the purported *production* environment.

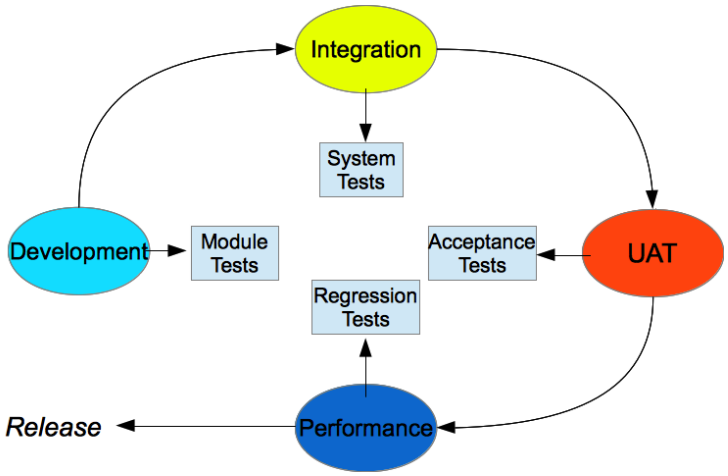

Figure 12.2: Simplified sequence of SW development and test phases

12.1.1 PMBoK: Software development, tests, and integration

The **PMBoK** provides a description of *Work Units* as laid out in the *Work Breakdown Structure* **WBS** together with the structure of the teams in die *working groups*.

Here, the quality planning is realized according to the relevant **ISO** standards. Specific requirements for the integration phase, however are not detailed but in addition a dedicated *Configuration Management* is foreseen.

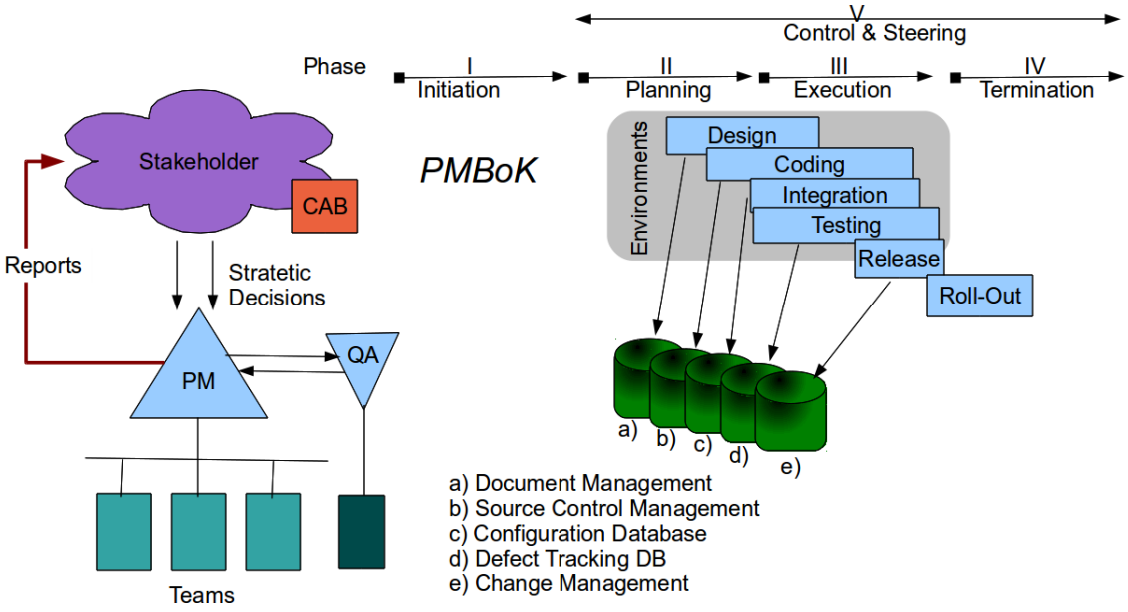

Figure 12.3: SW development according to PMBoK

12.1.2 Scrum: Software development, tests, and Integration

Within *Scrum*'s *Product Backlog* the *functional* and *none-functional* elements together with the *usage conditions* of the product/feature are included in the *User Story*, while the *detail design* is carried out within a *Sprints* from the team as port of it's operational *tasks*.

Due to the required 'mixed' competences of the *Sprint* team typically only *White Box* tests are considered. *Black Box* tests are subject of another *Sprint*. However, the team is responsible to realize and maintain it's own *Integration* environment.

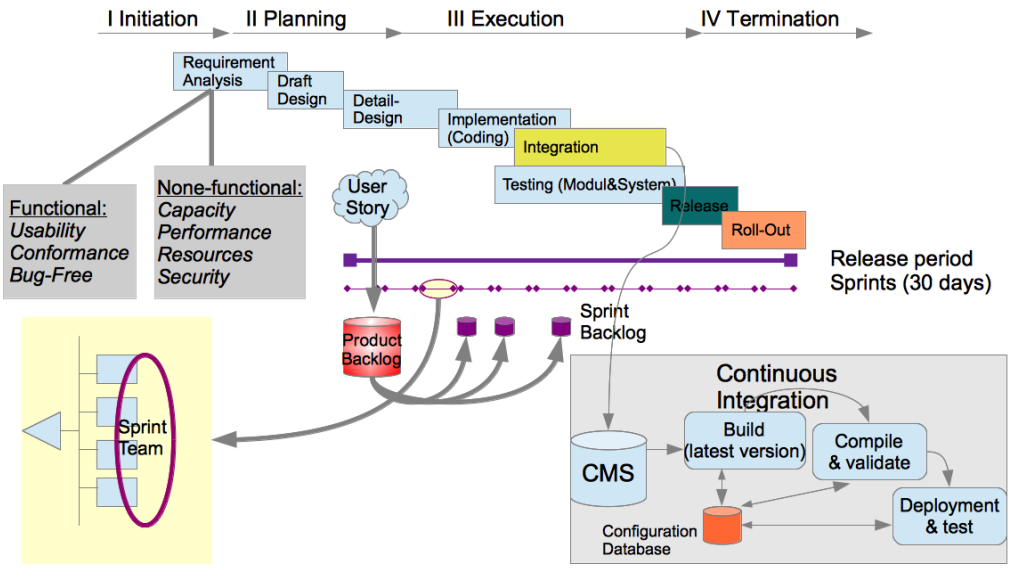

Figure 12.4: SW development according to the Scrum model

12.2 Quality Standards and Requirements

Quality standards for the *none-functional requirements* as part of *SW Engineering* have been investigated in since 1970, in particular by *Barry Boehm*, who tried to realize those by means of specific *SW development models*.

While the *functional requirements* can be precisely defined already in the SW design and realized while developing the SW and finally their implementation tested, deficiencies due to not caring about the relevant **NFR** are typically found much later, and perhaps too late (when the product is finally released).

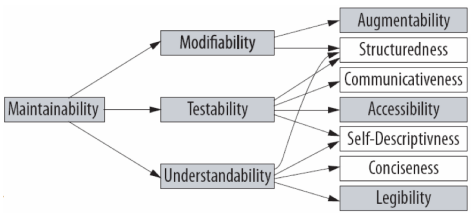

Figure 12.5: SW quality criterion's according to Barry Boehm

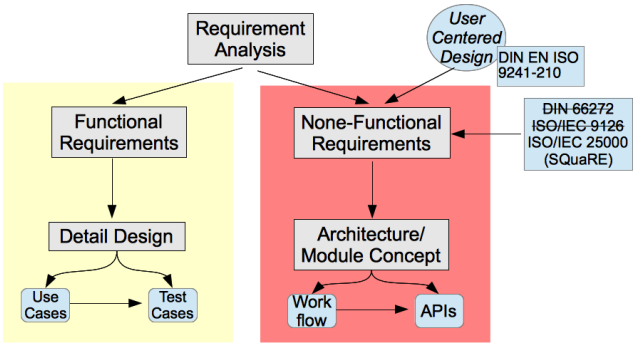

Figure 12.6: Rôle of FR and NFR in the development chain and attached quality standards

One approach is, to define the **NFR** more precisely; another is to care about their compliance in a *canonical* way already within the development chain:
Continuous Integration.

12.2.1 FURPS Criterion's

Software quality is often described accordingly to the *FURPS* criterion's:

FURPS: *Functionality, Usability, Reliability, Performance, Supportability*

The (withdrawn) standard DIN 66272 details those criterion's:

Functionality • Correctness • Interoperability • Compliance • Safety	Reliability • Maturity • Recoverability • Fault tolerance	Usability • Understandability • Learn ability • Operability
Efficiency • Responsiveness • Resource usage	Changeability • Analysable • Modifiability • Stability • Verifiability	Portability • Adaptability • Install-ability • Conformance • Exchangeability

Table 12.1: Bouquet of quality criterion's according to DIN 66272

This compilation (in particular the grouping) is not in accordance with our current understanding, thus DIN 66272 has been withdrawn in 1994.

12.3 SW Quality Management according to ISO 9000

The standard ISO 9000-1 questions software quality for 'Information Systems" IS in chapter A.?:

- Management attention: Is there any IT Manager for the IS accountable and responsible to define requirements and to approve changes?
- Quality Management System: Are all requirements for the IS explicitly laid down in documents?

- Audits: Are all requirements accompanied by a description and how to verify it's conformance?

According to ISO 9000 high-level management attention the most important factor required for quality management.

The quality management system QMS itself can be viewed from an

(a) descriptive and
(b) from an operational perspective

as outlined in figure [12.7].

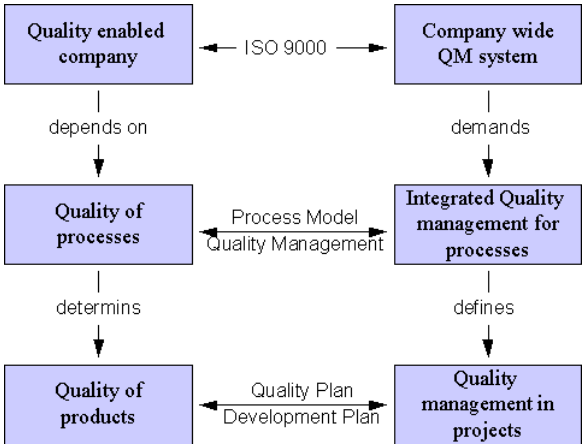

Figure 12.7: Set-up of a Quality Management System according to ISO 9000-3 [45]

The content of a **QMS** depends on the production branch described in the following ISO standards: QMS

- ISO 9001: Standard for quality management in design, development, production, assembly, and customer services.
- ISO 9002: Standard for quality management for production and assembly.
- ISO 9003: Standard for quality management for final testing and control.

Thus for software development, ISO 9001 is the required standard and demands continuous process improvements as shown in figure [12.8]:

The standard ISO 10013 provides in addition a lay-out of a Quality Management System (QMS) and emphasises the rôle of documentation in order to achieve conformance with this standard (figure [12.9]).

While setting up a QMS according to ISO 9001, in fact one has to include QM elements from the ISO standards 9001, 9002, and 9003. Thus, the final QMS handbook has to have the following scope:

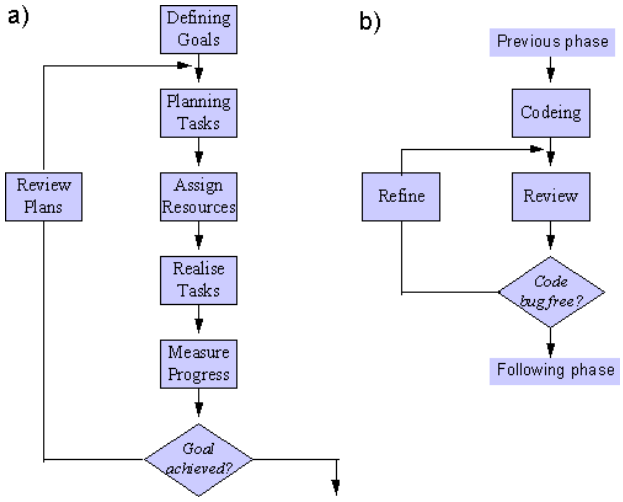

Figure 12.8: Quality Management for software development a) iterative process improvements regarding quality b) constant improvements for developments by means of testing [45]

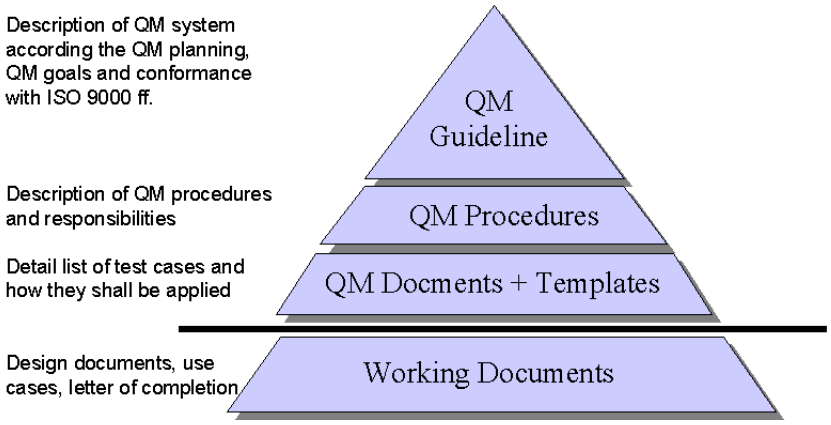

Figure 12.9: Hierarchy of a Quality Management System according to ISO 10013 [35]

Apart from assuring the quality management cycle for products, the conformance of the existing QMS has to be verified by means of Audits (ISO 9000-1). Audits are part of the ISO 9000 certification chain and may include:

- Content and conformance of the QMS handbook with the ISO standards.
- Operational conformance with the QMS handbook.
- Tests of the product quality.
- Capability of the project team to manage QM processes.

In Germany, some major organisation support the certification according to ISO 9000:

- Deutsche Gesellschaft zur Zertifizierung von Qualitätssicherungssystemen (DQS)
- Deutsche Gesellschaft für Qualität (DGQ)
- Technischer Überwachungs Verein (TÜV)

12.4 Quality standards according to ISO/IEC 9126

Following the standard `ISO/IEC 9126-1` a *hierarchical* quality model can be considered to consist from the following components:

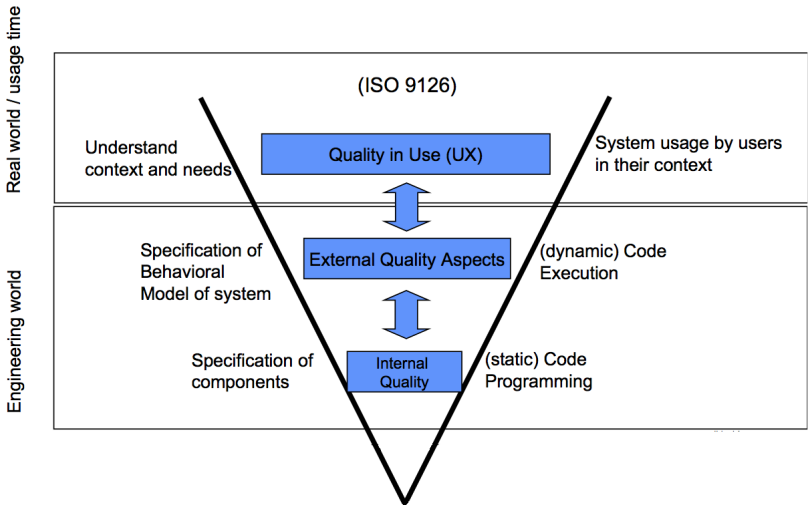

Figure 12.10: Definition of internal, external und quality in use according to `ISO/IEC 9126-1`

In particular the *Quality in Use* is s sensitive measure of the **NFR** compliance.

12.4.1 Quality as a Process chain – ISO/IEC 9126-1

The quality standard `ISO/IEC 9126-1` requires a particular *Process Model* thus the *defined* quality real reaches the *User*!

12.5 Software product Quality Requirements and Evaluation – SQuaRE

The current standard `ISO/IEC 2510` continues the ideas laid out in `ISO/IEC 9126-1` (2001) and possess the same view on *quality processes* and *quality definitions* but includes now explicit requirements for *quality measures*:

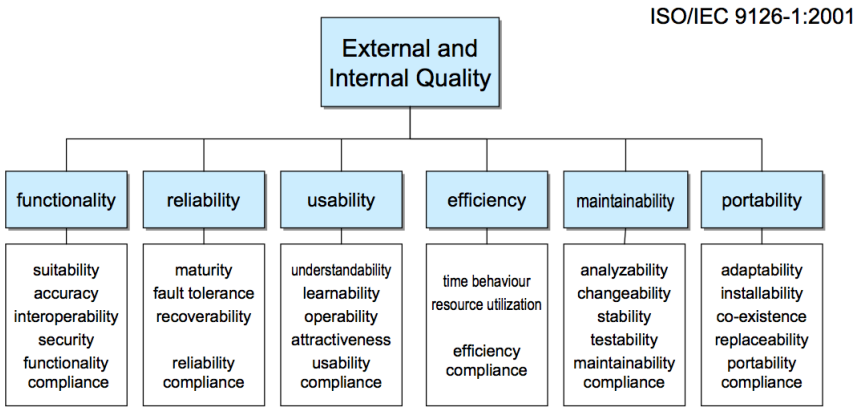

Figure 12.11: Internal and external criterion's according to ISO/IEC 9126-1

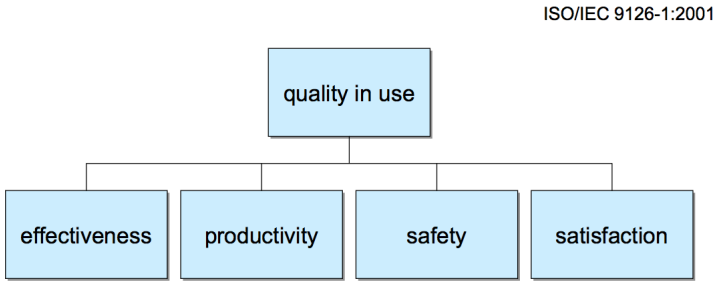

Figure 12.12: Quality in Use criterion's – ISO/IEC 9126-1

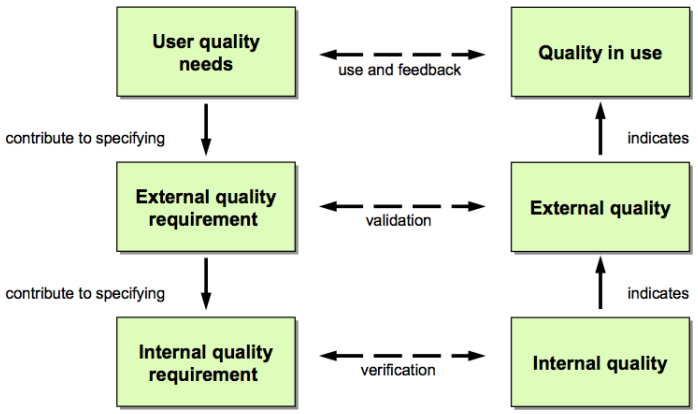

Figure 12.13: SW quality process chain – ISO/IEC 9126-1

- ISO/IEC 25010 2011-03 Software-Engineering
  Quality criterion's and Assessment of Software Products (SQuaRE) – Quality Model and Guideline
- ISO/IEC 25012 2008-12 Software-Engineering
  Quality criterion's and Assessment of Software Products (SQuaRE) – Model and Data Quality
- ISO/IEC 25020 2007-05 Software-Engineering

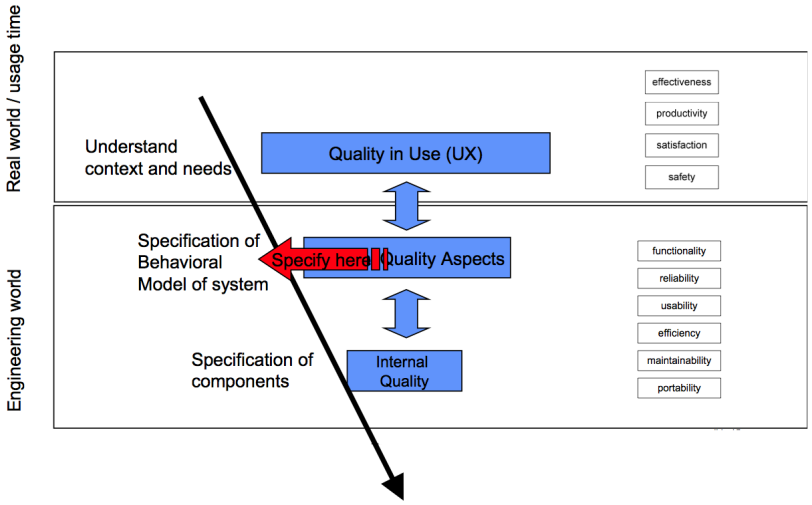

Figure 12.14: Quality criterion's of the components – `ISO/IEC 9126-1`

Quality criterion's and Assessment of Software Products (SQuaRE) – Quality Measurement – Measurement Reference Model and Guidelines

- `ISO/IEC TR 25021 2007-10` Software und System-Engineering
  Quality criterion's and Assessment of Software Products (SQuaRE) – Elements of the Quality Measurement
- `ISO/IEC 25030 2007-06` Software-Engineering
  Quality criterion's and Assessment of Software Products (SQuaRE) – Quality Requirements

The scope of this standard includes on only the quality of *software products* but now in addition the quality of *software systems* are detailed in addition with *usage criterion's* for commercial products.

12.6 Defect Management

As outlined,

a ***defect*** is a deviation of a software component from the documented behaviour and/or expected output.

12.6.1 Attributes of a Defect

For any defect, the following attributes can be assigned

- *Source* Source
  The reason for a Defect could be:
  - A programming error; commonly known as Bug.

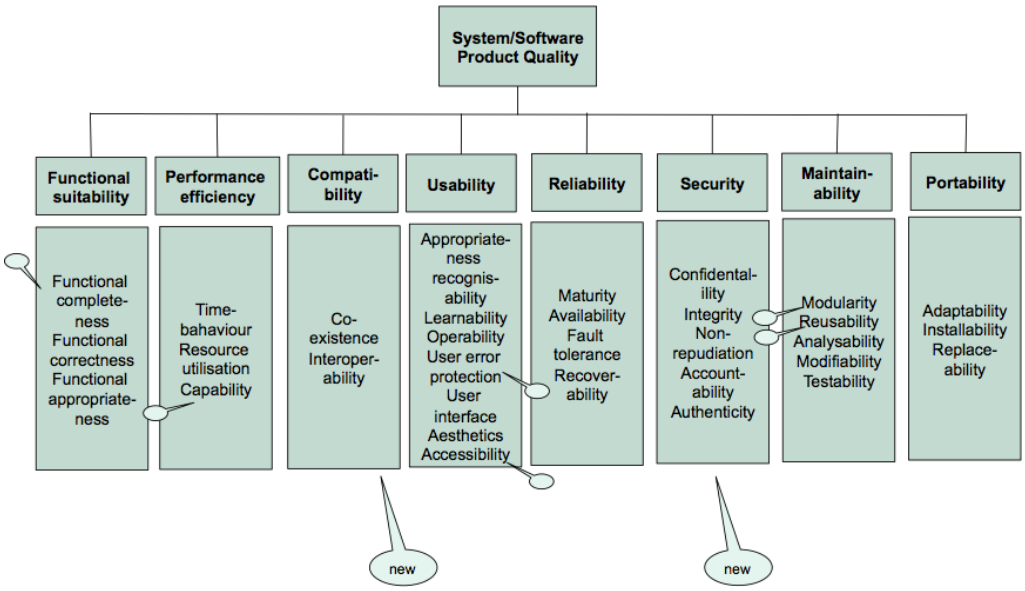

Figure 12.15: Quality criterion's according to SQuaRE

  - A programming context error; the anticipated (software) functionality works differently as documented/expected.
  - A design error; the programmer realised the code accordingly to the design, but this was inappropriate for the task.
  - A documentation error; the software component reacts differently with respect to the documented behaviour.

Priority

- *Priority*
  In our today's understanding, we use the following Priorities for the defect fixing:

  (-1) Showstopper; the defect needs to be fixed because it inhibits any further testing or any potential use of the software component.
  (1) The software component is inadequate or any use.
  (2) The software includes severe deficiencies and may lead to substantial deviations from the expectations.
  (3) The software shows deficiencies but they can be compensated (by work around).
  (4) Less-relevant errors have been accounted, which are not important for general use.
  (5) Errors have been accounted, but are not mainly due to insufficient documentation.

Category

- *Category*
  Any Defect is assigned to a component Category. Occasionally, a software module may include several components:

  - User Interface (Input / Output)
  - Middleware, Transport Layer, Interfaces/APIs
  - Back-end (e.g. Database)

  It is the responsibility of the software design to attach a certain component to category, suitable for Defect tracking.

- *Ownership* Ownership
  A defect is assigned for error-fixing to a particular Owner. In turn, every software is developed and perhaps trigger by a certain person-/team/ organisation/company. Thus, there should be a relationship between Ownership of the defect and the Authorship of the software component. Typically, Source Control Management Systems (SCM) and/or *Integrated Development Environment* (**IDE**) will automatically insert Author information taken from the user environment in terms of a header.
- *Versions* Versions
  Versionising the software component can be done explicitly by the developer or is automatically added by the SCM/IDE. Independent of the components version, it typically is developed for and available in a certain Release which provides a numerical (or verbal) identification of the whole project.
- *Project* Project
  A major software component or a set of components is typically identified as Project. Projects maybe subdivided in subprojects. However, this depends on the WBS (in PMBoK terms) and can be freely chosen in any software development project.
- *State* State
  The state describes the recognition of the defect (whether it is new, open or closed etc.) and it's assignment state, as discussed in the defect life cycle.
- *Due-Date* Due-Date
  The expected date, when the defect shall be fixed (this depends on priority).

12.6.2 Defect Lifecycle

For software development, essential part of the QM system is a bug-tracking or Defect Management software. One of the most-common systems is Bugzilla [47] (public domain). Typically, any Defect Management system uses a database as back-end and a graphical (ie. Web based) front-end. Logically, such a system allows us to define a Lifecycle for a defect. This Lifecycle can be forged to our own needs, or follows a standard procedure. In practice, companies may want to use one common system for Defect management and as well for Incident and Problem management, since the Lifecycle idea is the

same. Such Trouble Ticket Systems allow a Class definition for the occurred error to be reported:

- *Defects* – Software Development, Bug tracking
- *Incidents* – Deviations for a defined process (erratically behaviour); (none-) recurring
- *Problems* – Set of (inter-depending) incidents with (known) common source

In terms of Incidents and Problems not the Priority is of importance but rather it's Severity. Here, the correct routing (= assignment) of tickets is most relevant. Such systems may be extended by an *Artificial Intelligence* **AI** component with allows correlation of incidents, by time, location, and/or source. In this way, incidents may automatically sorted and forwarded and solutions strategies are proposed based on similar incidents or known (and solved) problems. The State of a defect will be changed and reflects the current actions on this defect. A minimal scheme could be the following:

Bugzilla

In addition, it might be necessary to qualify the solution (or in software development terms 'the fix') in some more detail. *Bugzilla* uses the following approach:

Transition changes among phases are not allowed to happen arbitrarily, but rather depend on the life-cycle model in place. This can be expressed in complex Defect State Charts as shown in figure [12.16]:

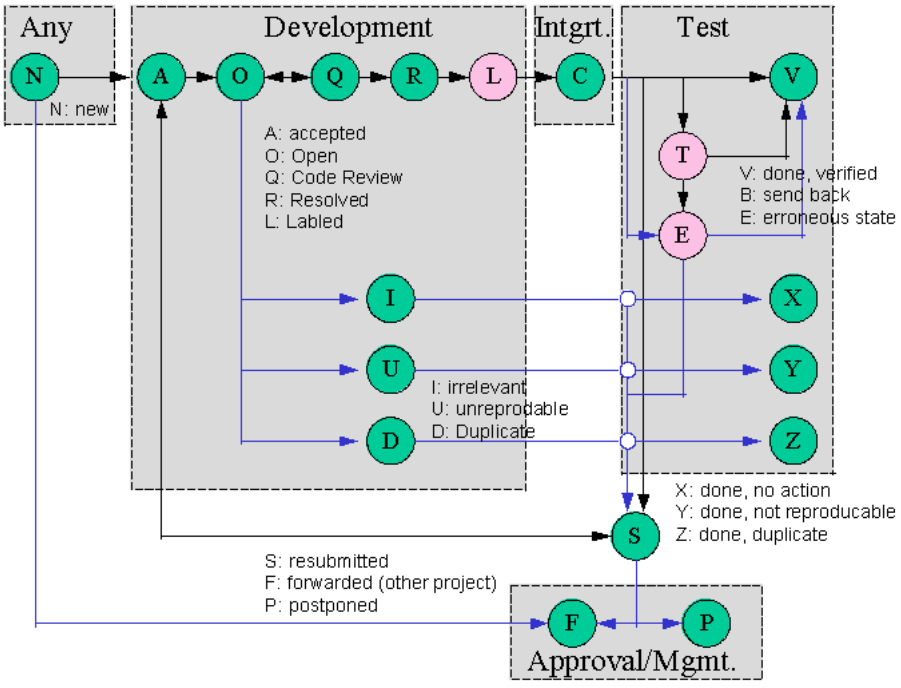

Figure 12.16: State chart for defects

12.7 QA reports

Quality Management or Quality Assignment Reports are generated regularly (for instance every week) and includes essentially tow different reports:

- a statistical report, representing the defects per project/subproject in terms of priority and component
- an individual report, focusing on the most important defects, while provide a short description of its current state and forthcoming solution strategies and due-dates.

Typically, a statistical reports is shown in terms of 'Lego charts' allowing a quick understanding of the project's defect distribution in terms of priority and component (figure [12.17]). In order to allow a quality measure, the following analysis shall be done:

1. How many defects of priority X have been fixed since last report?
2. How many new defects of priority X have open since then?

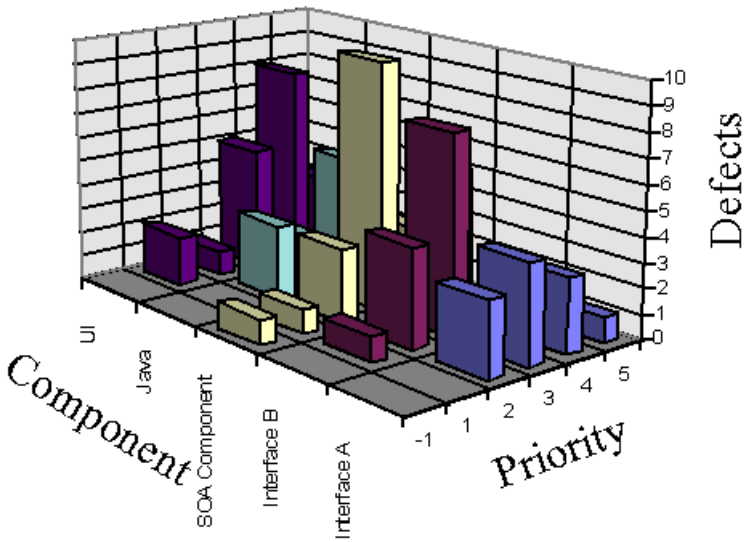

Figure 12.17: Defect distribution in terms of components and priorities

One essential task of the Quality Manager is to gauge individual events and bring them to attention. The following reasons could be considered:

- *Showstopper*: The defect impacts all other tests, since they depend on it's fix.
- *Criticality*: The defect inhibits the use of an important component and the solution is critical for the whole project.

- *Due-Date*: The solution for this defect has be postponed too often and disturbs other developments.

Together with the head of development, the Quality Manager will re-assess the defects and either require an intensified consideration of the defect or perhaps re-prioritise it. The head of development will then re-assign and re-schedule developers to potentially fix the defect.

12.8 Estimating remaining Defects

In particular in the advent of a forthcoming release, it is important to have a quantitative estimate of the number of potential open bugs in the product or perhaps per component. In case of qualified Quality Management system and under the assumption that defects have been treated in a controlled manner, we already have the following QA information:

- Distribution of the number of new defects in terms of priority and components.
- Distribution the number of fixed defects in dependency of priority and component.

What we don't currently have, is

- the number and distribution of unknown defects.

It is most common for software development to allow within a 'release' a number less severe defects. This in turn requires an estimate of the number of potentially remaining defects which have to be added to the number of known bugs. The key here is to use additionally development information:

- Phase 1: In the beginning of the software project, the established code base is small and bugs (even prio 1) happen often.
- Phase 2: While the development team becomes familiar with it's tools, the approach, writing a set common utility programs or classes, and gathering more and more experience, development becomes streamlined and the code base grows proportional in time and in numbers of developers. Watching this from the Quality Manager's perspective, quality increases and defects 'come and go'.
- Phase 3: However, coming close to a scheduled release, often development realises that completion is behind the original schedule. Thus, missing (but promised) functionalities have to be included in a rush. The code basis will probably increase significantly.

Figure [12.18] shows a sample, where the development plan assumes code completion happens accordingly to gaussian distribution, while real coding is deferred by some δ in time.

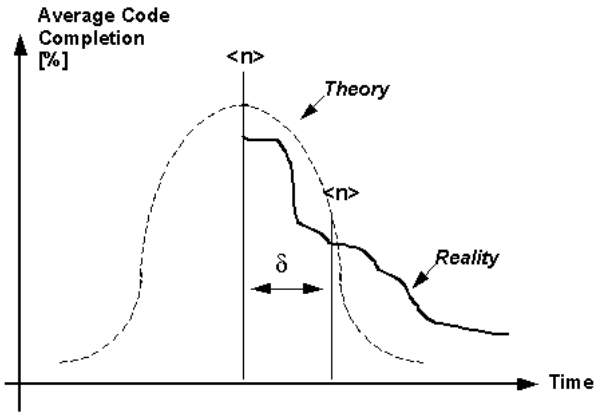

Figure 12.18: Average code completion as scheduled (dashed), as realised (solid)

In order to control quality and to consider the rapidly growing code base, the Quality Manager has to correlate the number of defects with the actual checked in code. For any software development it expected, that the number of defects depends on the lines of code produced, expressed as '*Defects/kLoc*' (number of defects per 1000 lines of code). Thus, even with most advanced QA means a certain number of bugs is be present and is acceptable. The Quality Management Plan (QMP) will probably detail, what is the amount of acceptable bugs and will provide a threshold (figure [12.19]).

kLoc

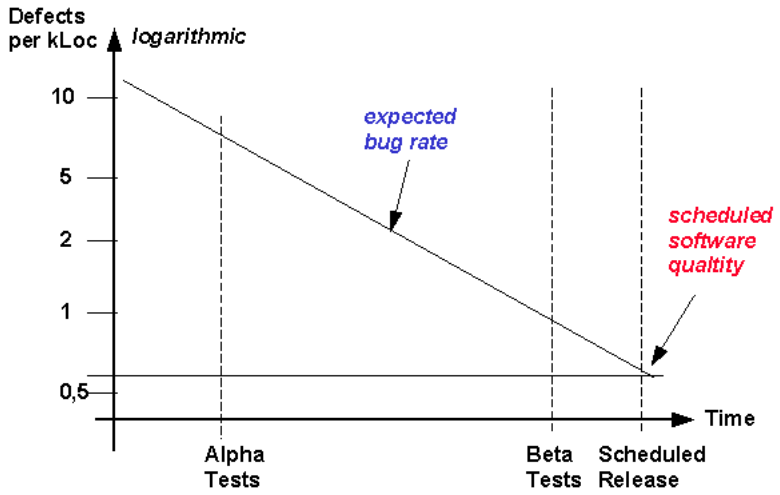

Figure 12.19: QMP chart with an estimate of acceptable defects for the scheduled release

Now, it is task of the Quality Manger to correlate the number of identified, fixed defects per *kLoc* and show this distribution on the same time line. A qualified extrapolating (not necessarily linear) will yield a guess of the number unidentified as shown in figure [12.19]:

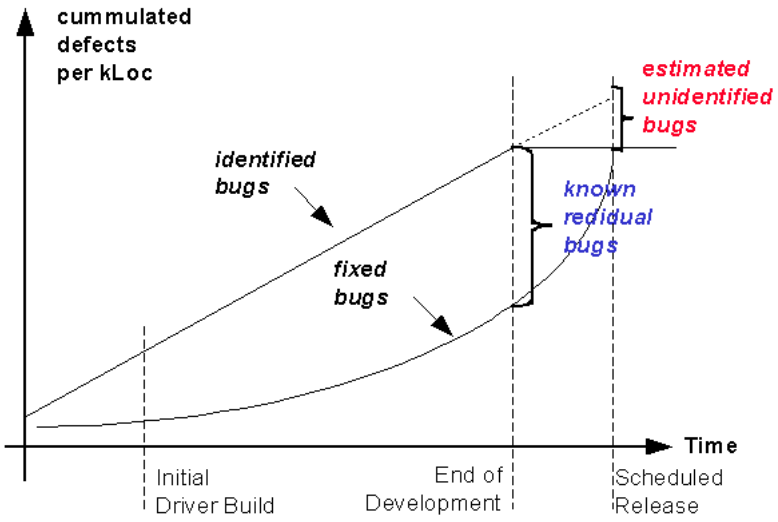

Figure 12.20: Estimation of unidentified defects for the final release

Chapter	**Title**	**Requirements**		
		ISO 9001	ISO 9002	ISO 9003
1.	Responsibility of Management	complete	complete	partially
2.	Quality Management System	complete	complete	partially
3.	Contract Verification	complete	complete	complete
4.	Design Management	complete	not available	not available
5.	Management of documents and code	complete	complete	complete
6.	Ordering Management	complete	complete	not available
7.	Management of third party deliveries	complete	complete	complete
8.	Labelling and Tracking deliveries	complete	complete	partially
9.	Process Management	complete	complete	not available
10.	Test procedures	complete	complete	partially
11.	Verification of test tools	complete	complete	complete
12.	Status of tests	complete	complete	complete
13.	Management of missing products	complete	complete	partially
14.	Correction and preventive means	complete	complete	partially
15.	Usability, packaging, and deployment	complete	complete	complete
16.	Management of quality reports	complete	complete	partially
17.	Internal quality audits	complete	complete	partially
18.	Education and training	complete	complete	partially
19.	Maintenance and customer service	complete	complete	not available
20.	Statistical means and procedures	complete	complete	partially

Table 12.2: Elements of a QMS document

Category	**Scope**	**Criterion**	**Methods**
internal Quality **iQ**	Static code	Correct + adequate data-model and workflow; name space; modularity; descriptive	Use cases; coding standards; design; quality audits.
external Quality **eQ**	Code execution	Bug-free; functional model; minimal dependencies	Test cases; module testing; QMS
Quality in Use **QiU**	Usage	Usability, productivity, safety, performance, maintainability	Release policy; defect management

Table 12.3: Quality categories and their impact

Defect State	**Meaning**
Unconfirmed	The defect has been reported but has not been checked successfully
New	The defect has been reported, but yet not assigned and/or verified
Assigned	The defect has been assigned and forwarded to a responsible developer
Reopened	The defect was handled and closed, but requires further investigation/treatments
Resolved	The defect has been solved and the solution requires approval
Verified	The defect is solved and the solution was verified
Closed	The defect is solved and the solution is integrated

Table 12.4: State of a defect according to Bugzilla [7]

Fix State	**Meaning**
Fixed	A bug-fix has been applied
Invalid	The defect was no due to a bug; further information is required
Wontfix	The defect can't be solved under the current conditions (time, budget)
Later	The fix for the defect is deferred (next release/version/update)
Remind	No fix will be provided now, but considered for a forthcoming version
Duplicate	The defect is a duplicate of another one
Worksforme	The defect can not be reproduced in the developer's environment

Table 12.5: State of a defect fixing according to Bugzilla [7]

Chapter 13

Continuous Integration

13.1 Why (Continuous) Integration ?

Lessons learned:

- In particular regarding 'big' projects, the amount of code develops not continuously, but rather in significant steps ('the due date problem').
- Also, SW modules which were initially planned to be be developed independently are recognized to be (tightly) coupled and providing significant dependencies:
  - One developer checks code out from repository, modifies it and checks it in and commits.
  - Other developer from different groups do the same with their code-base without realizing the dependency.

Checkins & Commits

As a result, the entire code diverges and at the end of the development period, no common compilation is possible → Showstopper !

Solution:

- *Continuous Integration* is required to understand and disentangle those dependencies at the earliest stage.
- Following the approach of periodical '*nightly builds*' the code modules are checked for consistency.
  - In case the code is able to be compiled at a whole and validated/tested in a further step, development can carry on.
  - However, facing problems while compiling or validating the code, the dependencies can be early identified and corrections may take place.

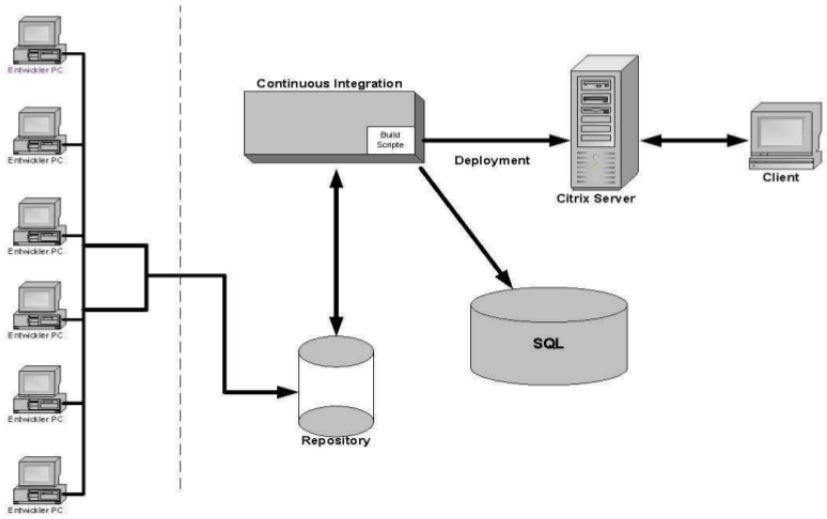

Figure 13.1: Procedure of Continuous Integration

This is accomplished by automatic *build tools* known as 'Hudson' and 'Maven'.

Instead of the 'nightly builds' the code can be compiled and validated 'on-the-fly' at every commit of a developer.

13.2 Development and testing Environments

In order to support the software quality management cycle, as part of the **QMP** different environments are usually used and referred to. Now, what is a environment? If common resources, including operating system, middleware (database), and other required applications a dedicated and configured for a dedicated task, we can refer his as environment. Typically, the following environments are used and configured for larger software developments:

DEV
- *Development environments*:
  This includes in particular any required compiler and linker, Integrated Development Frameworks, and Source Code Control Systems. At least one specific development environment is required for Release Management.

UAT
- *User Acceptance Test environments*:
  The (system) test environment's reflect closely, but on a smaller scale the later production environment; specific tools for software tests and quality management may be available. Here, the final quality approval will be carried out.

INT
- *Integration Test environments*:
  In case third party systems need to be included, one particular UAT can be used as Integration Test environment.

- *Reference environments*: REF
  This environment should be sized and configured comparable to the production environment. Here, performance and regression tests can be performed. Occasionally, the reference environments can be used as backup and fail-over systems for the production.
- *Production environments*: PROD
  Full sized production environment. Different from UAT, additional hardening could have taken place. Their impact has to be figured out in the reference environment.

13.2.1 Infrastructure for the Continuous Integration

In order efficiently use *Continuous Integration* the developer team (or parts of it) needs to set up an own *CI* environment, consisting of the following system components:

- The *Content Management System* **CMS** taking care of the project's source code. Whether a central repository (like **SVN**) or a 'virtual' distributed database à la *Git* is used, does not matter. CMS = VCS
- The particular **CI** server, interacting with the **CMS** while fetching those pieces of the code labelled to be used for *Integration*.
  The *Integration* is facilitated by mean of *build tools*. These may be simple scripts (for instance the **C** *Makefile* utility) or complex and powerful SW tools, to be used mainly for Java projects, like *Maven* or *Jenkins*. The **CI** is additionally used to generate the *reports* as a result of the *builds* and tests. Build tools
- The *Configuration Management Data Base* **CMDB**, which describes and manages the code pieces and the accompanying *parametrisation*. CMDB
- A *Web server* used to visualize and deploy the results to the developer and in particular the quality manager.

13.3 Requirements for the Team applying Continuous Integration

In order to setup *Continuous Integration* in automatic manner, the team is required to prepare the input of the input of the **CI** system:

- The *SW developers* need to check in the code into the **CMS** with significant information:
  - Any used *public function* needs to described.
  - The *name space* conventions in order to declare functions, classes and variables are obligatory.

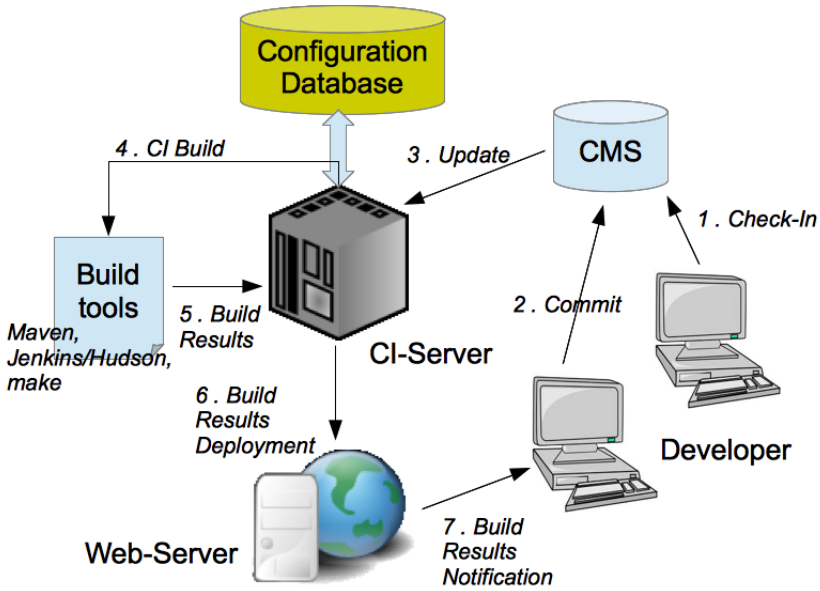

Figure 13.2: Process chain of Continuous Integration

  - The coding *style guides* need to be followed.
  - The size and complexity of functions shall not exceed a critical limit.
  - A useful *internal* description of the SW components is required, which can be extracted automatically.
  - The developers need to check-in additionally the *Makefiles* to compile their components.

- The **CI Server** managers are responsible to streamline and monitor the automation process:

  - Technical problems of the *build process* need be be detected, analysed, and fixed.
  - The response of the *build process*, the *build state* needs to be communicated in a verbose way to the developers.
  - The *CI* staff members support the developers in case of *build breaks*.
  - The *build results* are documented and forwarded to the quality manager.

Additionally, the SW developers are required to create together with the **CI** managers *automated tests* which allow to verify the principal fitness of the compiled modules.

13.4 The Build Process

The developers will maintain and develop the code according to the releases. In a VCS, the developers can maintain a 'view' of the particular releases. Software modules are checked in and checked out against a certain release. Thus we have to consider two distinct logical views:

- The *Branch* – this is a logical order software modes in the software repository, the VCS, common for all users. Branch
- The *View* – this is the workspace of the developer; a view may consist of different branches. View

It is the requirement of the Release Manager to determine when the current branch (see figure [13.3]) should be forked, thus a new (development) release is due. This decision impact the consistency of the code base: Release Manager

- In case of a too early branch, often changes in the old branch have to applied to the new branch.
- In case of a too late branch, development capacities are still focused on the old release, thus the new release may be delayed.

Branching

Release Management has to realise, that each release (=branch) has its own SDLC, thus generally, any branching should be avoided. Of course, this binds resources and causes efforts, perhaps not planned initially.

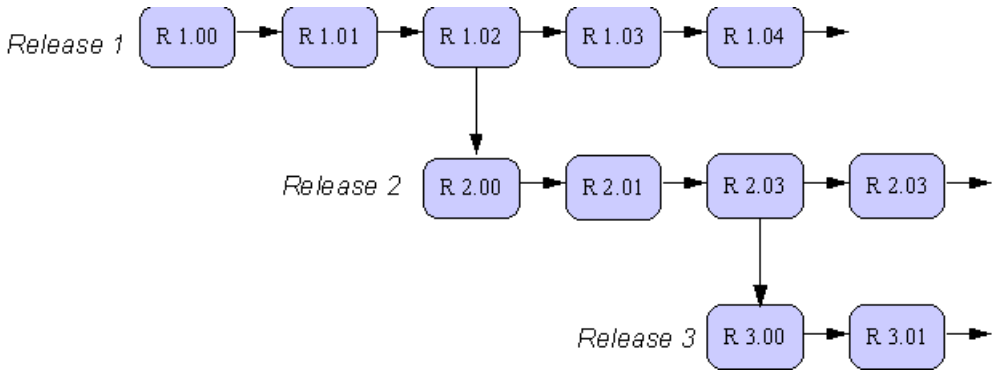

Figure 13.3: Branching into different releases in the VCS [20]

As a consequence, the developer may occasionally modify modules in a wrong branch, since this information is typical not visible for the developers, since they are assigned to a particular release/branch. It is the obligation of the Release Manager or head of software development, first to communicate with the developers regarding the release schedules and second to correct wrongly checked-in modules (figure [13.4]).

Labelling

All software components which should be included in the release are *labelled*. Effectively, even modules can be labelled for a certain release, which belong to a different branch.

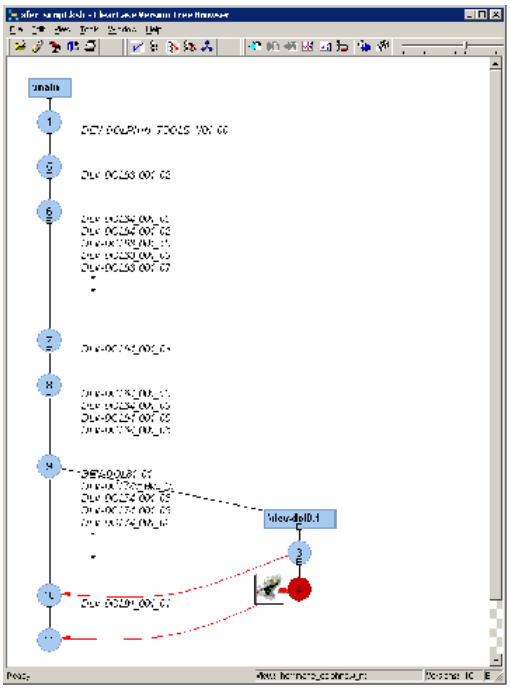

Figure 13.4: Including a changed module into a specific branch

Merging

tbd.

13.4.1 Building for Continuous Integration

As already mentioned, the *build process* might be either run

- *continuously* and been automatically triggered by the *commitment* of a module, or
- as '*nightly build*' at the end of a developer's day.

The result of a *build* could be one of the following cases:

1. Success: Compilation and the automatized tests have been completed with no error/warning.
2. Warning: During compilation errors have been encountered, or same validation steps could not been passed successfully.

3. Incomplete: The **CI** build process did not finish in time; perhaps because of the size of code or the dependencies, which were needed to disentangle in the first place (*configure*).

4. Error: Due to critical errors the **CI** build failed $\Longrightarrow$ Build Break!

Build Breaks delay the entire development process and impact the entire development/quality process substantially.
After a *Build Break* has been realized, no new code shall be committed until the reasons have been identified and the **build** process was recovered.

13.4.2 Fixing errors within Continuous Integration

Since the problems of Microsoft's project 'Longhorn' (aka *Windows Vista*) became public known, *Continuous Integration* is in the focus of development/quality management for SW projects.

In case of problems during the *build* process, a particular strategy is required:

- *Private Builds*: In this case, not the entire system is build, but rather a certain developer or developers group's component are subject of the *Integration* test, what is called *Pre-Test Commits*.
- *Broken Builds*: Now the developer team needs to put it's entire attention to identify hat piece of (new or changed) code which triggered the *build* failure. Probably a *Roll-Back* to on older version might be helpful.

- *Mockups*: Instead of the error-prone components, *dummy routines* called *Mocks* or *Mockups* can be used instead to at least finish the *Integration* tests.

13.5 Continuous Delivery

Continuous Integration is an important step towards the final *release* of the SW product, the *deliverable* respectively:

- In addition to the integrated, automatized tests, further (external) quality tests according to the *Quality Plan* **QP** are required.
- Typically, adjacent *Alpha* and *Beta* tests are scheduled.

Is the SW product subject to be delivered to the *product owner* once the *CI build* was successful, we call this *Continuous Delivery*.

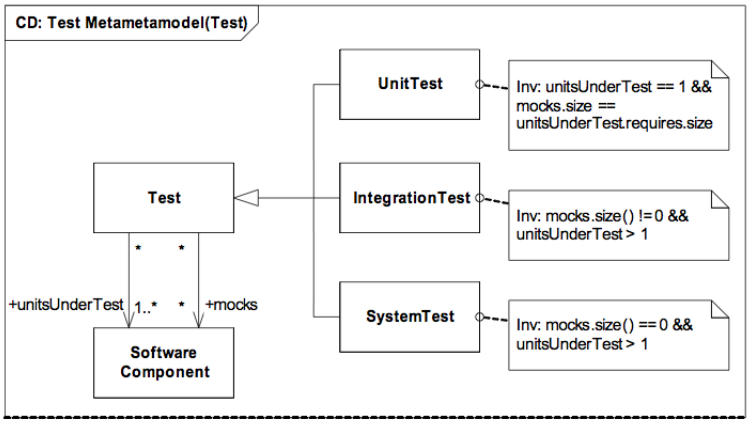

Figure 13.5: Use case for using *Mockups* instead of the real code

Chapter 14

Release and Roll-out Management

Given an IT Software Project, we can consider Release Management to be that management discipline which is closest visible for the customer – and impacts him most important. While preparing this script a 'race' is ongoing about the next generation Web Browser among the combatants Microsoft's IE7, Firefox 3, and finally Opera 9.5 (Apple's Safari does not play a relevant rôle under Windows). This order reflects the market shares, however, the release order is inverted, and reflects probably the pressure from the marketing department (figure [14.1]). Despite of this Release and Roll-Out Management is not directly covered by any Project Management Framework, however it is defined as **ITIL** *Service Management* process. Often, Release Management is considered as part of Change Management, though this does not cover the complexity of the Release Management process entirely.

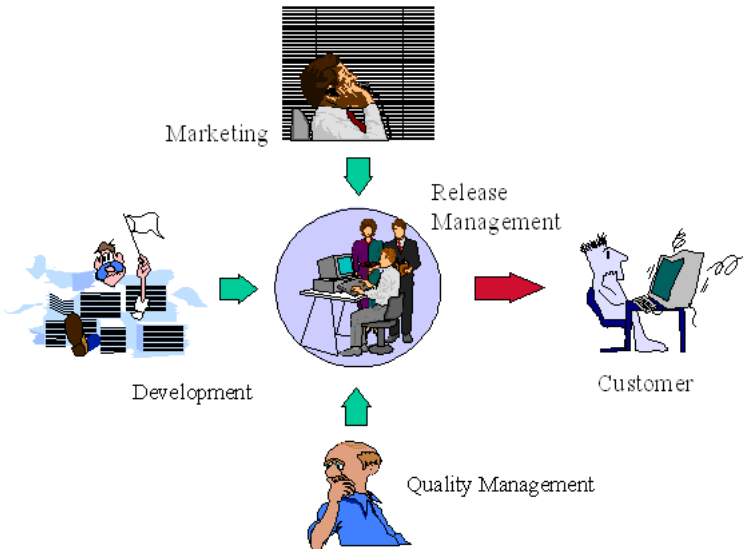

Figure 14.1: Release Management as key discipline for customer satisfaction

14.1 Release Management à la ITIL

SMF

Release Management is of part of ITIL's (IT Infrastructure Library) **Service Management Functions** as shown in figure [14.2]. According to ITIL's understanding Release Management is required for

- the introduction of substantial and critical hard changes,
- the deployment of relevant software,
- bundling complex and interdependent changes.

The Release Management has the task of co-ordinating the Service Providers, delivering companies, and other involved third parties in order to integrate all products subject for deployment and complement them with the required documentation and perhaps training.

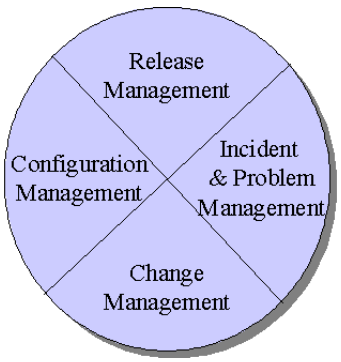

Figure 14.2: ITIL Service Support Management Functions (SMF)

A central concept of ITIL is the *Definitive Software Library* . This serves as repository for the code, while configuration parameters are stored in the *Configuration Management Database* . Note, that the DSL is not equivalent to the database of our *Source Code Management System* and the **CMDB** is not the same as our *Document Management System*. The release process has to be guided by some principals of conduct. In particular, the rôles of the different involved parties (figure 88) have to be clearly defined and the responsibilities of the Release Management have to be expressed (see figure [14.1]).

DSL

CMDB

VCS

DMS

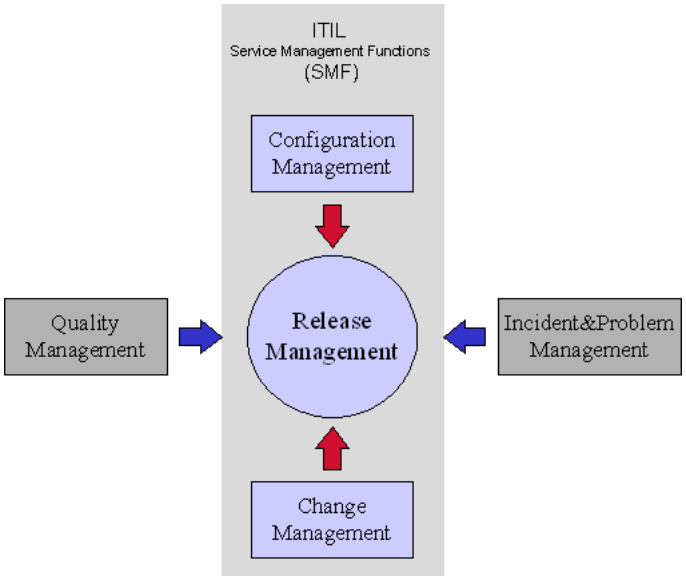

Figure 14.3: Dependencies of the Release Management with other management disciplines

Regarding ITIL, the main tasks of the Release Management are:

- Planning and supervision of roll-outs considering any changed hard- or software, including all relevant documents.
- Alignment with Change Management regarding the contents of the release and a detailed rôle-out schedule.
- Responsibility that all products subject for release are available and are registered as entries in the CMDB.
- Providing release information for customers in order to prepare the acceptance of the products and to shape the expectations.

Concerning the forthcoming release, ITIL recognises the following release-types:

- *Full Release*: All components have been commonly developed, tested, deployed and implemented.

Full Release

Delta Release

- *Delta Release*: Only particular changes, encountered since the last release are considered .

Package Release

- *Package Release*: Is a release superset, including different products (even third party) and full or delta releases.

ITIL does not consider software development and quality management explicitly in it's functional model, however makes the following recommendations:

- Build Management
  The soft- and/or hardware subject for a release have to 'build' in a consistent manner, allowing a reproducible processes. Some processes have to be automated in order to improve consistency and to minimise human mistakes. As soon, as the test-phase starts, Build Management should be the responsibility of Release Management.
- Test Procedures and Fall-Back Strategies
  Any release has to be carefully tested, prior of the approval, the user acceptance has to be verified. In addition. Fall-back strategies and plans have to be prepared, documenting what steps are required, in case the release and/or the final deployment is error-prone. The fall-back strategies and plans have to be tested in the reference environment.

14.2 Release Management Overview

Alternatively to ITIL's approach within Project Management we can consider Release Management as part our quality approach and thus belongs to Quality Management (see figure [14.3]). While Quality Management interfaces with development, Release Management interfaces mainly with the customers.

Figure [14.4] shows the dependencies between Release Management, Software Development, Quality Management, and Change Management. We identify, that Release Management is an essential part of the *Software Delivery Life Cycle* .

SDLC

CAB

- Change Advisory Board
  Typically, the Release Manager is member of the *Change Advisory Board* **CAB**. More complex releases (i.e. an Operating System) will probably employ different Release Managers for the different products subject to release. Other members of the CAB are the Quality Manager(s), the head of Software Development, and perhaps representatives (bridgeheads) of other companies, in case of third party software/hardware.
- Software Development
  While software development can be facilitated according to the mod-

els as discussed in the previous chapter, the Release Manager has finally to identify each software component to be included in the final release. Probably within the software development department, somebody maintains the content of the *Source Control Management System* . The last step of the development process is to label in the VCS the individual components and modules. The Release Manager and the head of Software Development have the common responsibility to ensure the completeness and the consistency of the release.

VCS

- Quality Assurance
  The main part of the quality management process belongs to the software development phase or stage. However, the Quality Manager plays an import rôle to guarantee the absence of major bugs in the release. Unfortunately, a 100% bug-free delivery is almost impossible and thus the Quality Manager has in the advent of a forth-coming release already to prepare resources to support post-release bug-fixing.

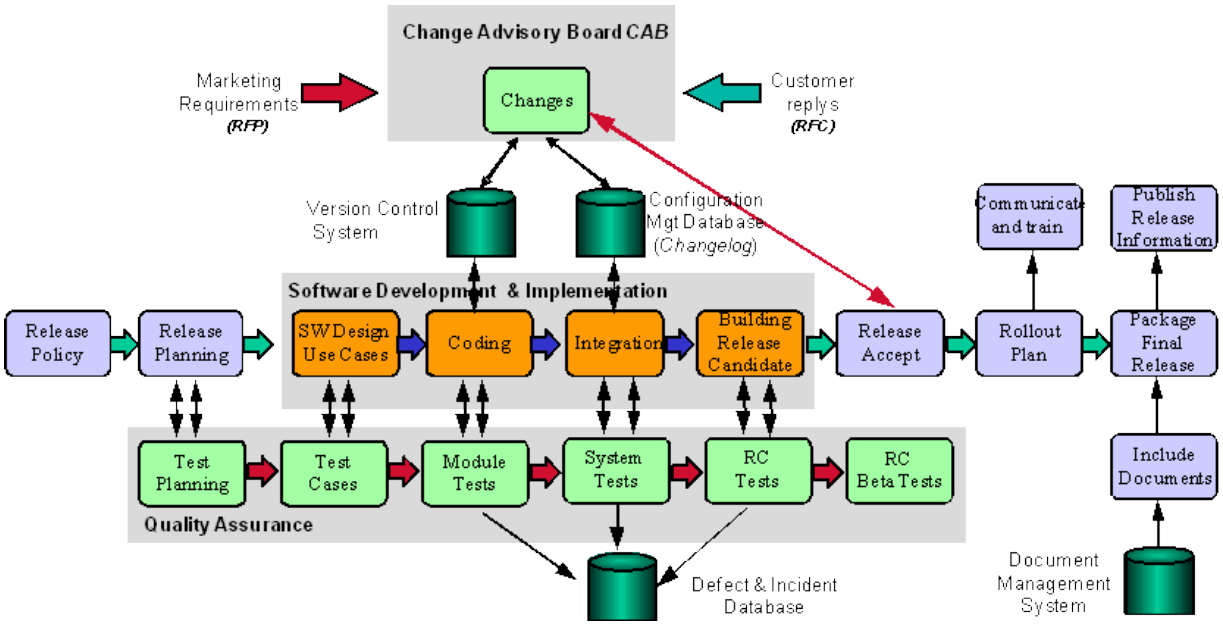

Figure 14.4: Release Management steps from design to final release

A special case has to be considered if the software product will be deployed on different systems, ie. levels of the OS or other potentially impacting causes. Here, software development and quality management can decide to use a controlled field test, typically known as 'beta tests' in contrast to the in-house 'alpha tests'.

14.3 Release Planning

Release Planning is different with respect to the initial release and the forthcoming releases. The release initiators and the context for next release in addition with the release request type are summarised in table :

Initial Release:

Even before the first modules are coded, Release Management should define how releases are identified and how the releases are realised in the VCS. Thus choice of the VCS and it's capabilities determine to some extend the flexibility of the release process. The Identification of a release can be quite difficult and is a little bit arbitrarily. Initially we have the following information:

Version
- The *Version* of the different software modules; either provided the VCS or by the software author him/herself.

Release
- The Version(s) and perhaps **Release**(s) of external components which have to bundled with the current release.

Build Number
- The *Build Number* of the VCS which labels a particular configuration or counts the steps for compilation/linking, thus to generate the executable modules.

CMDB
For efficient defect tracking all these information have to be included in the *Configuration Management Database* and are now commonly referred to as 'Release'. In order to support the SDLC, we typically use the following hierarchical scheme:

Major Release
- Major Release

Minor Release
- Minor Release

Fixlevel
- Fixlevel

and assign a number to a particular release `2.7.18`.

As discussed above, a *Minor Release* could be realised as *Delta Release* and will only install against a particular *Major Release*. Occasionally, this

V.R.F
is also called a `V.R.F.` scheme, with `V = Version`, `R = Release`, and `F = Fixlevel`. In this way, we can preserve the logic and the dependencies of the software development.

In addition, it might be useful to include a *Delivery Information* as well. In some cases, the delivery information is appended by a dash '-' or underscore '_' sign. For instance a release could be introduced as 1.2.0_02, telling the customer that this is the second ('02') delivery of release `1.2.0`. While the code base shall be the same for 1.2.0 and 1.2.0_02, maybe because some packaging errors occurred, or because extra documentation has been added a first and second delivery is necessary.

Rôle	**Context**	**Release Entry Point**	**Release request type**
Senior Executive	Discussions of largest grained needs. Final association of most serious issues ideation of exploration.	Senior IT Leader	Usually personal interactions and unstructured documents (emails, presentations, etc.).
			Request for information (RFI).
			Starts with personal interactions and moves into process driven.
	Requests for major new systems. Large project level.	IT CRM	RFP - Request for Proposal.
Unit Executive	Requests for major new systems. Large project level.		RFC (high level) Request for Change.
		Demand Management System	Starts as process-driven demand request or RFC
Functional area owner	Requests for new systems, additional functionality	Demand Management System	Starts as process driven demand request or RFC.
		IT Service Request System	Development requirements.
User	Requests for orderable IT Services Reporting of Incidents	Incident Management System	Process driven for service requests, reporting requests, Incident reports, or Change RFCs (operational)
		Change Management System	Management system. New system requirements.

Table 14.1: Releases Sources and Initiators [13]

Further, some companies tend to include the *Build Number* in the release information (i.e. for the Windows NT OS while preserving a descriptive name for the development). Other companies use additional names to a specific

release (i.e. Apple for the OS X - Panther: 10.3, Tiger: 10.4, Leopard 10.5).

Further Releases:

The release planning for the forthcoming releases is mainly triggered from four sides:

- *Market*: Marketing has identified a set of requirements and 'trends' to follow.
- *Customers*: The customers report defects and express need for additional functionalities.
- *Quality Management*: The open defects in the previous release needs additional fixes.
- *Technical Progress (Development)*: Changing standards have to be incorporated into the next release.

Here, we have to consider two main processes:

RFP
- **RFP** – *Request For Proposal*
  will typically include an initial definition for additional requirements

RFC
- **RFC** – *Request for Change*
  are issued by the Change Manager and need approval by the C*hange Advisory Board* **CAB**. All realised changes need to be documented. Prior for the next development round the *Change Advisory Board* has to digest the requested changes for the product and finally decide what to include into the next release. Changes have to be identified by module and impact. Apart from the sizing, they are also prioritised.

Thus, the release plan will include a development timeline, where changes are well defined for the next releases but will become more and move vague.

14.4 Code and Patch Management

Code and Patch Management is a task of Software Development, however during a release build, the Release Manager needs to audit this process. Typically, it is the software developer who modifies the code on it's own behalf. In particular for Open Source projects, code changes will also arrive by external developers or users. In addition, field engineers may change the code on customer demand, in particular to fix a problem. Changes to the source code or the executable module which modify the execution are called Patches. In case the patch is realised against the executable program, it needs to be re-engineered and included into the streamline.

Code changes and patches which impact the functional behaviour of any module (and thus do to fix a defect) have to consider four different issues:

- What is the reason for the code change ?
- Who is the maintainer of the code ?
- To which version of the software do we need to apply the change, thus in what release should the enhancement/change be incorporated ?
- Does this change impact Use and/or Test Cases ?

Figure [14.5] shows the inclusion of patches into the development cycle until final inclusion.

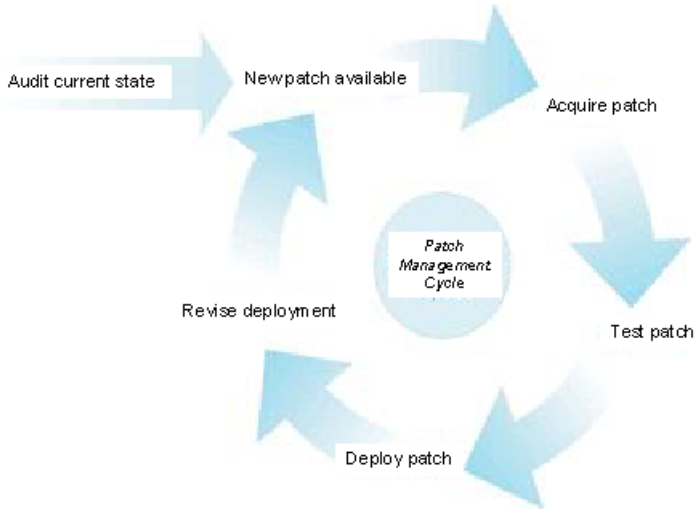

Figure 14.5: Patch Management Cycle [13]

14.5 Quality Assurance

The strong dependencies between Release and Quality Management can be depict from figure [14.6].

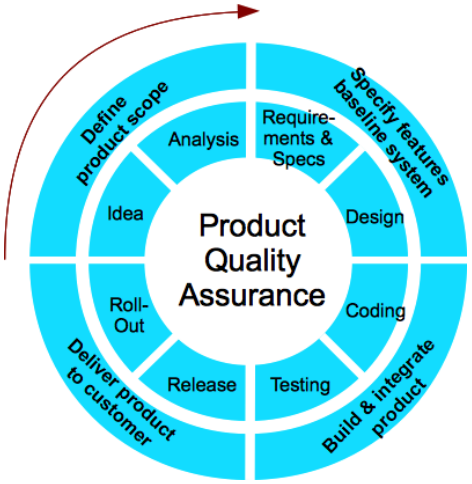

Figure 14.6: Quality Assurance from the idea to the release

As discussed, both the Quality Manager and the Release Manager have to be prepared about the remaining defects in the current release. Figure [14.7] shows on a time line the 'creation' and fixing of defects. By means of a good quality management system, most defects can be corrected before the actual product comes into operation. Most probably, fixes for the remaining defects are scheduled for the next (minor) release. However, it might be necessary to deploy important bug fixes in a particular *Hotfix* release. There exist different opinions about *Hotfixes*. Occasionally, they are part of the standard release cycle or they are real 'hot-fixes' required for special customer problems/issues. Important is the recognition of a hot-fix for further enhancements. Typically, hot-fixes are 'stand-alone', where as in particular any minor release will recognise the existing installation base and update this respectively.

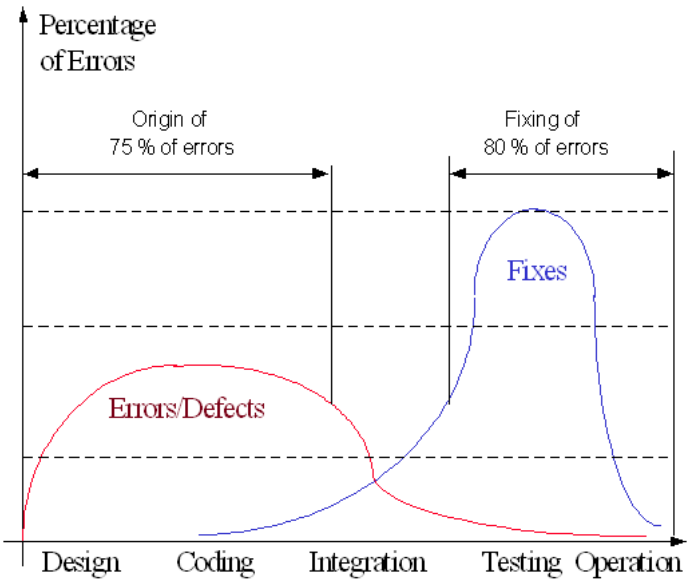

Figure 14.7: Defect appearance and fixing during the development and test cycle

14.6 Release Readiness

Declaring Release Readiness is responsibility of the Release Manager. However, the final approval is done by the Change Advisory Board. Depending on the current state of completion and defects, the CAB may conclude to issue a Release Candidate (RC) prior of the final release (figure [14.8]). The Candidate build may be issued to:

- the QA department
- the public or interested customers/users.

While the QA can determine the readiness according to a well-defined environment, a 'public RC' does not; however it can be used to gather experience and perhaps to identify design problems, not recognised yet.

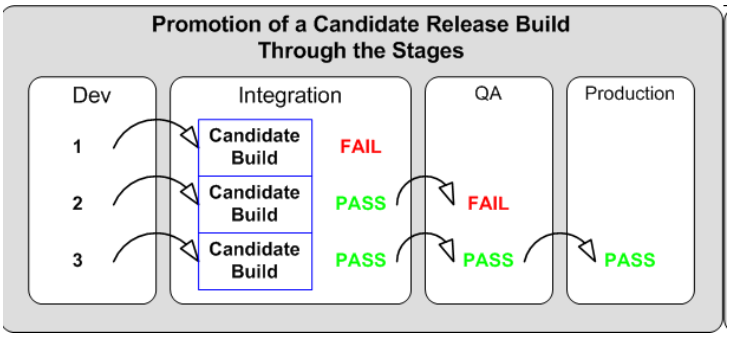

Figure 14.8: Building Release Candidates

14.7 The Roll-Out Process

The Roll-Out process may include several software and even hardware products. For example, today's OS are couples with email-systems, web-browsers, text-editors, multi-media-software and other 'gadgets'. Thus within the roll-out process different products need to be bundles as part the Release Management. Additionally, the Roll-Out process has to include the required release documents. Depending on the product and the targets may include:

- Installation documentation
- Release documentation
- User manuals
- Administration guides

Once the roll-out is prepared, the medium for the roll-out has to be set-up. While 'in the old days' software releases are shipped on diskettes or CDs, nowadays, everyone accepts downloads from a particular Download Server. In any case, it has to be verified by the Release Manager, hat the required resources are available

Chapter 15

Summary

The previous chapters covered most of today's understanding of IT Project Management. We have seen, that a few general Project Management Frameworks do exist, in particular PRINCE2 and PMI's PMBoK which are generally used and well suited for the management of IT Project. In addition, IT Projects use IT specific tools which are generally not covered by the general frameworks.

While most aspects of project management can described and successfully applied by the means discussed in the previous chapters, the risk management is certainly the most challenging part of the operative project management.

While there is a general discussion, whether project management is a part of a general risk management, or risk management is a specific aspect of project management, regarding IT project two different 'incarnations' of risks have to be considered and dealt with:

1. Intrinsic risks due to the new technology (*applicability*).
2. Risks due to missing quality (*usability*).

While IT provides to means to estimate and reduce (2.), the intrinsic risks can only be covered by 'traditional' project management; thus needs a qualified Project Manager and this ability to 'sense' those risks and to cope with them.

15.1 The Project Management Timeline

Close to the PRINCE2 approach, I have subdivided the IT project into four phases:

Figure [15.1] shows the particular tasks assigned to the different phases (without claiming to be complete) in terms of 'project devil's two-sided fork'. We see, that during the execution phase, the two particular management aspects become vital:

	Phase	**Project tasks**	**Product development**
1	Initiation	Setting up & circumstances	Idea & scope
2	Planning	Project planning	Requirement analysis, draft design
3	Execution	Operation & Controlling	Implementation, testing, integration
4	Termination	Closure, decommissioning	Release, roll-out

Table 15.1: The four phases of project management and SW development

- *Operative Management* (day-by-day management)
- *Management by Controlling*

15.2 Linking Project Management with SW Product Development

In the previous chapters I have detailed the inter-relationship between general *Project Management* with the particular *SW Development* and it's requirements.

Typical for *IT* projects is a substantial amount of *Product Management*:

- Requirement Analysis
- Specification
- Design (high and low level)
- Implementation (coding and integration)
- Testing (module and system)
- Release and perhaps
- Roll-Out

Companies focused on projects with a high degree of SW development have tailored their 'production chain' to cope with these conditions. A central component is the *Quality Management* touching both *project management* and *product development* [15.2]

This complex environment results in addition into a dedicated task sharing of responsibilites between several (Project) managers, which is common sense for larger projects.

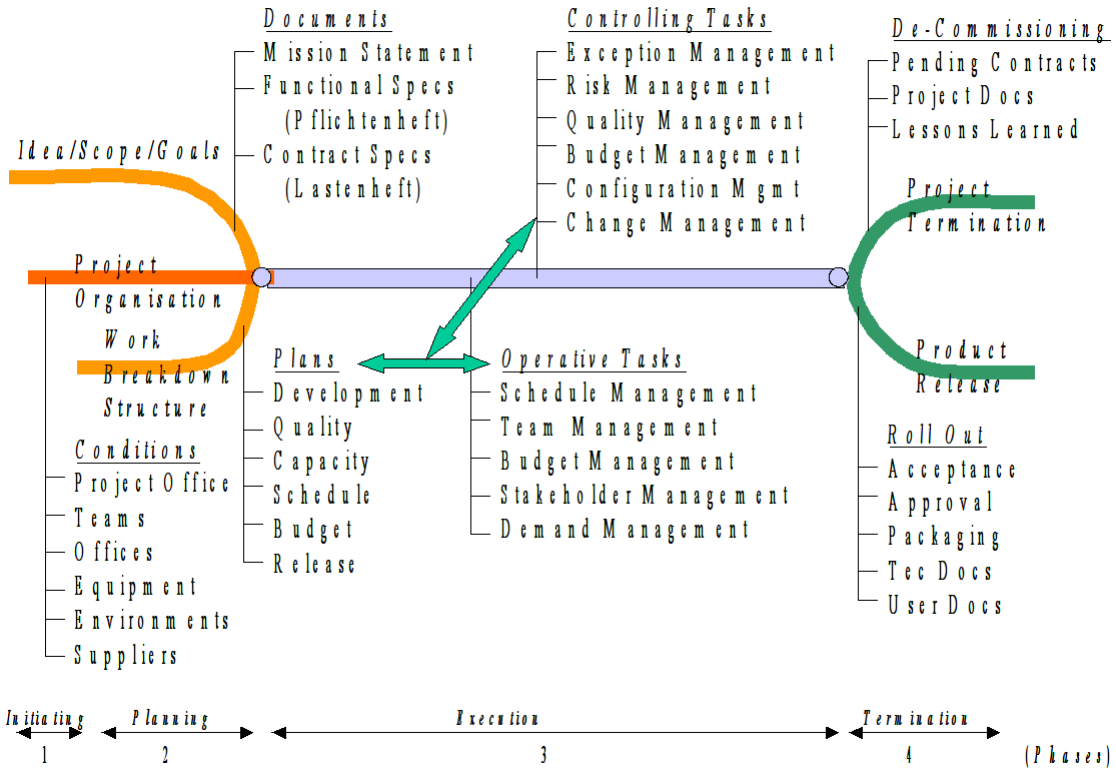

Figure 15.1: 'Project devil's two-sided fork'

15.3 Open Issues

While the further chapters covered the issues of IT Project Management to some detail, still some missing subjects have to be addressed:

- *Software Design and Implementation*
  I only mentioned modern paradigms of software design and development briefly. Currently discussed are in particular 'design patterns', among others. Many of those issues only become relevant in a particular development environment, typically the the context of OO and Java development.
- *Programme Management*
  The coordination of several ongoing projects needs a particular steering which is called 'Programme Management' and/or perhaps 'Portfolio Management'. Though within the PRINCE2 approach Programme Management is mentioned, in practice a lot more specific requirements need to meet.

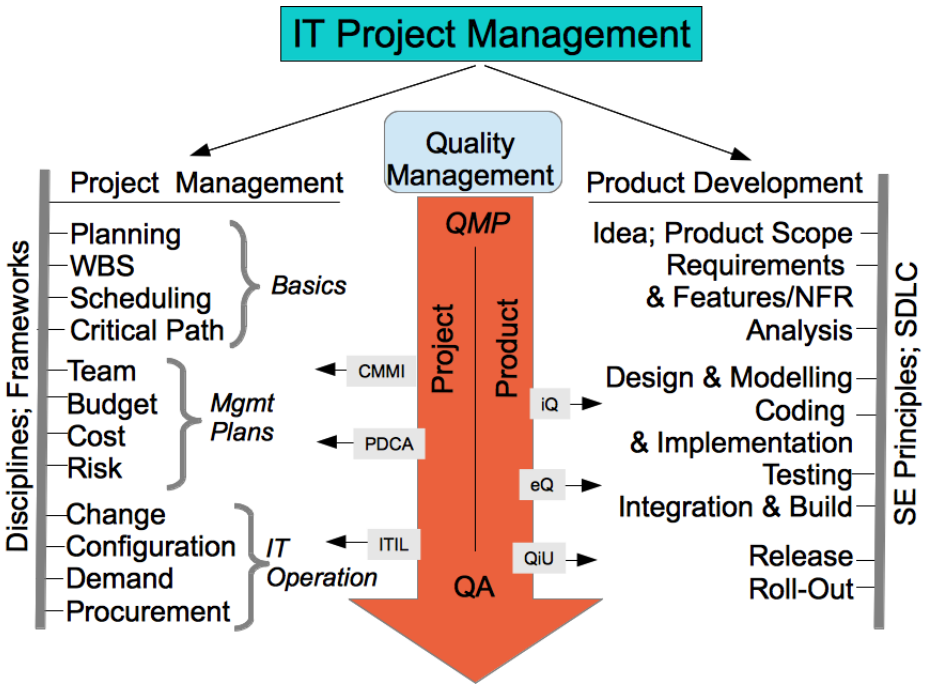

Figure 15.2: The principal ingredients of IT Project Management; iQ: *internal Quality*, eQ: *external Quality*, QiU: *Quality in Use*, NFR: *None-functional Requirements*, QMP: *Quality Management Plan*, QA: *Quality Assurance*

Chapter 16

Project Artifacts

16.1 Project Initiating Statements

Project basic facts	
Project's name:	
Name of Project Manager:	
Stakeholder of Project:	
Accountable/Approved by:	
Estimated duration of Project:	
Expected Milestones:	

Table 16.1: Project's Mission Statement

Business Statement	
Is situation:	Problem statement,
Outcome:	Achievable aims; current, proposed
None-aims of project:	out-of-scope issues
Expected usage:	For whom ? What ? How far ? Strategic ?
Impact in case of failure:	

Table 16.2: Project's Business Statement

Ressource Estimation			
	Human Resources Expenditures (days and Euro):		Operational Expenditures (OpEx):

Rer

.

.

.

.

Table 16.3: Resource Estimation

Chapter 17

PMBok English/German Glossary

Abbreviation	**English Term**	**German Terminology**
AC	Actual Cost	Ist-Kosten
ACWP	Actual Cost of Work Performed	Ist-Kosten der geleisteten Arbeit
AD	Activity Description	Beschreibung des Vorgangs
ADM	Arrow Diagramming Method	Vorgangspfeilnetzplan
AE	Apportioned Effort	Zugeteilter Aufwand
AF	Actual Finish Date	Tatsächlicher Endzeitpunkt
AOA	Activity-on-Arrow	Vorgangspfeilnetzplan
AON	Activity-on-Node	Vorgangsknotennetzplan
AS	Actual Start Date	Tatsächlicher Anfangszeitpunkt

Abbreviation	**English Term**	**German Terminology**
BAC	Budget at Completion	Ursprünglich geplante Gesamtkosten
BCWP	Budgeted Cost of Work Performed	Fertigstellungswert
BCWS	Budgeted Cost of Work Scheduled	Budgetkosten der geplanten Arbeit
BOM	Bill of Material	Stückliste
CA	Control Account	Kontrollkonto
CAP	Control Account Plan	Kontrollkontenplan
CCB	Change Control Board	Steuerungsgremium für Änderungen
COQ	Cost of Quality	Qualitätskosten
CPF	Cost-Plus-Fee	Selbstkostenbasis plus Honorar
CPFF	Cost-Plus-Fixed-Fee	Selbstkostenbasis plus Pauschalbetrag (Werkverträge)
CPI	Cost Performance Index	Kostenentwicklungsindex
CPIF	Cost-Plus-Incentive-Fee	Selbstkostenbasis plus Leistungshonorar (Werkverträge)
CPM	Critical Path Method	Methode des Kritischen Wegs

Abbreviation	**English Term**	**German Terminology**
CPPC	Cost-Plus-Percentage of Cost	Selbstkostenbasis plus prozentualer Kostenanteil (Werkverträge)
CV	Cost Variance	Kostenabweichung
CWBC	Contract Work Breakdown Structure	Vertragsgegenständlicher Projektstrukturplan
DD	Data Date	Datum des aktuellen Stands
DTA	Decision Tree Algorithm	Entscheidungsbaum-Struktur
DU/DUR	Duration	Dauer
EAC	Estimate at Completion	Erwartete Gesamtkosten zum aktuellen Zeitpunkt
EF	Early Finish Date	Frühester Endzeitpunkt
EMV	Expected Monetary Value	Erwarteter Geldwert
ES	Early Start Date	Frühester Anfangszeitpunkt
ETC	Estimate to Complete	Erwartete Restkosten zum aktuellen Zeitpunkt
EV	Earned Value	Fertigstellungswert
EVM	Earned Value Management	Management des Fertigstellungswertes
EVT	Earned Value Technique	Fertigstellungswertmethode
FF	Finish-to-Finish	Endfolge

Abbreviation	**English Term**	**German Terminology**
FF	Free Float	Freie Pufferzeit
FFP	Firm-Fixed-Price	Festpreisbasis
FMEA	Failure Mode and Effect Analysis	Fehlermöglichkeits und Einflussanalyse
FPIF	Fixed-Price-Incentive-Fee	Festpreisbasis plus Leistungshonorar
FS	Finish-to-Start	Normalfolge
IFB	Invitation for Bid	Ausschreibung
LF	Late Finish Date	Spätester Endzeitpunkt
LOE	Level of Effort	Unterstützungsfunktion
LS	Late Start Date	Spätester Anfangszeitpunkt
OBS	Organizational Breakdown Structure	Organisationsorientierter Strukturplan
OD	Original Duration	Ursprüngliche Dauer
PC/PCT	Percent Complete	Fortschrittsgrad
PDM	Precedence Diagramming Method	Vorgangsknotennetzplan
PF	Planned Finish Date	Geplanter Endzeitpunkt
PM	Project Management	Projektmanagement
PM/PL	Project Manager/Leader	Projektleiter

Abbreviation	**English Term**	**German Terminology**
PMBOK	Project Management Body of Knowledge	Project Management Body of Knowledge
PMIS	Project Management Information System	Projektmanagement Informationssystem
PMO	Program Management Office	Programmmanagementbüro
PMP	Project Management Professional	Project Management Professional
PO	Project Office	Projektbüro
PS	Planned Start Date	Geplanter Anfangszeitpunkt
PSWBS	Project Summary Work Breakdown Structure	Übersichtsprojektstrukturplan
PV	Planned Value	Geplanter Wert
QA	Quality Assurance	Qualitätssicherung
QC	Quality Control	Qualitätslenkung
RAM	Responsibility Assignment Matrix	Verantwortlichkeitsmatrix
RBS	Resource Breakdown Structure	Einsatzmittelstrukturplan
RBS	Risk Breakdown Structure	Risikostrukturplan
RD	Remaining Duration	Verbleibende Dauer
RFP	Request for Proposal	Angebotsaufforderung

Abbreviation	**English Term**	**German Terminology**
RFQ	Request for Quotation	Angebotsanfrage
SF	Scheduled Finish Date	Geplanter Endzeitpunkt
SF	Start-to-Finish	Sprungfolge
SOW	Statement of Work	Leistungsbeschreibung
SPI	Schedule Performance Index	Terminentwicklungsindex
SSD	Scheduled Start Date	Geplanter Anfangszeitpunkt
SS	Start-to-Start	Anfangsfolge
SV	Schedule Variance	Terminplanabweichung
SWOT	Strengths, Weaknesses, Opportunities and Threats	Stärken, Schwächen, Chancen, Risiken (SWOT-Analyse)
TC	Target Completion Date	Vorgegebener Abschlusszeitpunkt
TF	Target Finish Date	Vorgegebener Endzeitpunkt
TF	Total Float	Gesamte Pufferzeit
T&M	Time and Material	Zeit und Material (Dienstleistungsverträge)
TQM	Total Quality Management	Total Quality Management
TS	Target Start Date	Vorgegebener Anfangszeitpunkt
VE	Value Engineer	Wertgestaltung

Abbreviation	**English Term**	**German Terminology**
WBS	Work Breakdown Structure	Projektstrukturplan (PSP)

Table 17.1: PM Glossary [41]

List of Figures

List of Tables

Bibliography

[1] 12MANAGE. http://www.12manage.com/.

[2] Scrum Alliance. Scrum alliance. http://www.scrumalliance.org.

[3] Ch. Bartelt, Th. Ternité, and M. Zieger. Modellbasierte entwicklung mit dem v-modell xt. http://www.sigs-datacom.de/fileadmin/user_upload/zeitschriften/os/2005/05/bartelt_ternite_OS_05_05.pdf, 2005.

[4] K.J. Bechler and D. Lange. *DIN Normen im Projektmanagement*. Beuth Verlag, 2004. DIN-Taschenbuch 226: Qualitätsmanagement-Verfahren.

[5] Y. Bernard. Seminar erfahrungen und experiment im software engineering. 2005.

[6] B.W. Boehm. A spiral model of software development and enhancement. http://csse.usc.edu/csse/TECHRPTS/1988/usccse88-500/usccse88-500.pdf.

[7] Bugzilla. Bugzilla. http://www.bugzilla.org/.

[8] M. Cohn. Task board. http://www.mountaingoatsoftware.com/scrum/task-boards.

[9] William C.W. Scrum process mechanics. http://xp123.com/articles/scrum-development-on-a-page/.

[10] G. Diethelm. *Projektmanagement, Band 1: Grundlagen*. Verlag Neue Wirtschaftsbriefe Herne/Berlin, 2000.

[11] G. Diethelm. *Projektmanagement, Band 2: Sonderfragen*. Verlag Neue Wirtschaftsbriefe Herne/Berlin, 2001.

[12] Ch. Differding. Ein objektmodell zur unterstützung des gqm-paradigmas. 1993.

[13] M. Drapeau and S. Oudi. Release management: Where to start? http://www.itsmwatch.com/itil/article.php/3680776/Release-Management-Where-to-Start.htm, May 31, 2007.

[14] Tufte E. Powerpoint does rocket science–and better techniques for technical reports. http://www.edwardtufte.com/bboard/q-and-a-fetch-msg?msg_id=0001yB.

[15] eGroupware. egroupware. http://www.egroupware.org/.

[16] EPF. Artifact: Sprint burndown chart. http://epf.eclipse.org/wikis/scrum/Scrum/workproducts/sprint_burndown_chart_F647C347.html.

[17] Kent et al. http://agilemanifesto.org/.

[18] Standish Group. Chaos 2004 survey results. http://www.infoq.com/articles/Interview-Johnson-Standish-CHAOS.

[19] IBM. Ibm lotus notes. http://www-01.ibm.com/software/de/lotus/.

[20] Neuma Technology Inc. Taking the complexity out of release management. http://www.neuma.com/Neuma/ReleaseManagement.pdf.

[21] Project Management Institute. *A Guide to the Project Management Body of Knowledge*. CCTA, 1999.

[22] Software Engineering Institute. http://www.sei.cmu.edu/.

[23] B.W. Karnighan and D.M. Ritchie. *The C programming language*. Prentice-Hall, Eaglewood Cliffs NJ, 1978.

[24] H.-D. Litke. *Projektmanagement*. Carl Hanser Verlag, München, 5. erweiterte auflage edition, 2007.

[25] BASE Consulting LLC, Knowledge Systems Design (KSD), and Siemens ITS. System engineering guidebook for its, version 2.0. http://www.fhwa.dot.gov/cadiv/segb/files/segbversion2.pdf, 2007.

[26] R.L. Martino. Project management and control, vol. 1, finding the critical path, new york. 1964.

[27] Microsoft. Microsoft - sharepoint 2010. http://sharepoint.microsoft.com/en-us/Pages/default.aspx.

[28] B. Oesterreich. *Objektorientierte Software Entwicklung*. Oldenbourg Verlag, München Wien, 1998.

[29] Office of Government Commerce. Itil v3. http://www.ogc.gov.uk/guidance_itil.asp.

[30] Office of Government Commerce. Prince2. http://www.ogc.gov.uk/methods_prince_2.asp.

[31] Office of Government Commerce. *Managing Successful Projects with PRINCE2*. Newton Square PA, 4th edition, 2004.

[32] Spiegel Online. Kölner u-bahn-bau-affäre. http://www.spiegel.de/panorama/0,1518,681611,00.html.

[33] J. Phillips. *IT Project Management*. McCraw.Hill/Osborne,Emeryville CA, 2nd edition, 2004.

[34] P. Reich. *Die Massenpsychologie des Fashismus*. Matrix Verlag, 2005.

[35] R. Retrash. *Einführung in das Software-Qualitätsmanagement*. Logos Verlag, Berlin, 1998.

[36] Westdeutscher Rundfunk. Der kölner u-bahn-bau. http://www.wdr.de/themen/panorama/koeln/ubahn_stadtarchiv/ubahn/uebersicht.jhtml.

[37] J. Scherer. Scrum-fibel. http://www.scrum-fibel.de/.

[38] A.T. Schreiner. Was ist eigentlich unix ? (the essence of unix). 1/1987.

[39] W.A. Scott. A manager's introduction to rational unified process (rup). http://www.ambysoft.com/unifiedprocess/rupIntroduction.html.

[40] Softonic. Easy timesheets. http://easy-timesheets.en.softonic.com/.

[41] Alby T. Projektmanagement: Definitionen, einführungen und vorlagen. http://projektmanagement-definitionen.de/glossar/.

[42] A. Thomas. Erfahrungsbericht – methoden in der anwendungsbetreuung, September 29, 2004.

[43] Tigris. Tigris.org - open source software engineering tools. http://tortoisesvn.tigirs.org.

[44] Timios. Gnatt designer. http://timios.net/Gantt/more.html.

[45] E. Wallmüller. *Ganzheitlichen Qualitätsmanagmement in der Informationsverarbeitung*. Hanser Verlag, 1995.

[46] Wieczorrek and Mertens. *Management von IT-Projekten*. Springer Verlag Berlin Heidelberg, 2 edition, 2007.

[47] Wikipedia. Bugzilla. http://en.wikipedia.org/wiki/Bugzilla.

[48] Wikipedia. Capital expenditure. http://en.wikipedia.org/wiki/Capital_expenditure.

[49] Wikipedia. Cheops. http://de.wikipedia.org/wiki/Cheops.

[50] Wikipedia. Critical path method. http://en.wikipedia.org/wiki/Critical_path_analysis.

[51] Wikipedia. Function point. http://en.wikipedia.org/wiki/Function_point.

[52] Wikipedia. Ibm rational unified process. http://en.wikipedia.org/wiki/Rup.

[53] Wikipedia. Kölner dom. http://de.wikipedia.org/wiki/Koelner_Dom. URL adjustment: 'ö' was translated to 'oe'.

[54] Wikipedia. Operating expense. http://en.wikipedia.org/wiki/Operating_expense.

[55] Wikipedia. Plain old documentation. http://en.wikipedia.org/wiki/Plain_Old_Documentation.

[56] Wikipedia. Sarbanes–oxley act. http://en.wikipedia.org/wiki/Sarbanes--Oxley_Act.

[57] Wikipedia. Scrum. http://de.wikipedia.org/wiki/Scrum.

[58] Wikipedia. Spice. http://en.wikipedia.org/wiki/SPICE.