Project Management Guide

14th edition – January 2017

This guide is used as material for the Project Management course delivered in English at EPITA as part of its international Master programs.

Table of Contents

0) Foreword

This document, which I call a "guide", explains what a project and project management consist of. It describes the functions of a project manager, as well as the fundamental principles and basic rules of project management. Many of these principles and rules apply to any kind of project, but the focus of this guide is on projects concerning the creation and publishing of software, including multimedia applications and websites.

This guide is intended for future project managers, junior project managers, and, more generally, for anyone who needs to understand and deal with project management.
It addresses not only the technical aspects of projects but also other factors to be taken into consideration such as business models, human resources, suppliers, contracts, communication, the relationship with Marketing & Sales...
It covers all facets of the role of a project manager as well as many of those of a product manager. It also addresses aspects of management in general.

The content of this guide draws on the experience, including 25 years of management, I acquired in the course of my career, working for 12 years at Honeywell-Bull, 10 years at Apple and 12 years at Hachette, then, as of September 2007, as a freelance Project Management consultant for companies involved in digital technology and/or development & publishing, such as Ganaxa, IDM, Infostance (now itslearning), Kayentis, Fronter.

As director of a products division at Hachette, I managed, "from A to Z" so to speak, the entire 12-year-long life cycle of the *Hachette Multimedia Encyclopedia* (in French: *Encyclopédie Hachette Multimédia,* aka "*EHM*"). As co-designer as well as project director of the EHM, I played the dual role of "composer and conductor".

The chronology of the EHM extends from the project feasibility study phase, during the summer of 1995, to the final stages of the end of life of the product, from 2007 to 2010, including the multiple phases of content creation and enhancement, software (re)design and (re)development, and extension of the product from CD/DVD-ROM to the web.

The case of the EHM, to which I often refer in this guide, is particularly interesting since that product consisted of frequently-updated content (the encyclopedic assets, ie text and multimedia illustrations), of software (the "container"), on CD/DVD-ROM and on the Internet, of administration and editing tools (the "back office"), of packaging (box including a manual) for the CD/DVD-ROM, and, for the online versions, of a part free of charge and a part available on a subscription basis via a user authentication system.

Furthermore, the EHM project involved a broad variety of in-house as well as external contributors: authors, editors, proofreaders, data architects, software developers, graphic designers, testing, hosting and payment-system service providers, as well as marketing, sales, human-resources, legal and finance people…
That is why I have chosen the EHM project/product as a source of many of the "real-life" examples that are provided in this guide.

Neil Minkley (e-mail: neil@minkley.fr)

> Useful documents and links are available via **my Project Management web page**:

>> neil.minkley.fr/pm

> Also **visit my "hobby" site** for **vocabulary and sample sentences in French and English**:

>> anglais-pratique.fr (see in particular the "Gestion de projets" section)

> If you wish to **join my network on Linkedin and/or follow me on Twitter**, go to:

>> linkedin.com/in/neilminkley and/or @NeilMinkley

1) Introduction

What is a project?

Dictionaries provide various definitions of the word "project", for example…

Cambridge:

- "a piece of planned work or an activity which is finished over a period of time and intended to achieve a particular aim"

Chambers:

- "a scheme of something to be done; a proposal for an undertaking; an undertaking"

Oxford:

- "a planned piece of work that is designed to find information about something, to produce something new, or to improve something"

Webster's:

- "any piece of work that is undertaken or attempted; a planned undertaking"

Here is another definition which I have devised (for what it's worth!):

- A project is the set of actions leading from an idea to its concrete realization.

Finally, here are two definitions (from pmi.org and method123.com respectively) which are more precise and more technical:

- "A project is a temporary endeavor undertaken to create a unique product, service or result."

- "A project is a unique endeavour to produce a set of deliverables within clearly specified time, cost and quality constraints."

As compared with standard business operations or processes, projects have the following characteristics (as presented in method123.com):

- they are unique in nature, as opposed to being repetitive, each new project being different from previous ones;
- they have a defined timescale, including clearly specified start and end dates;
- they are assigned a specific, limited and approved budget, which is often treated as an investment;
- they have well-identified resources;
- they involve an element of risk, due to the fact that their end result is in the future;
- they are usually intended to achieve beneficial change.

A project may be simple or complex, short or long. Whatever its features, **a project must be managed** in order to produce the expected result.

What is project management?

To complement the above definitions of the word "project", here are the brief definitions of the words "**management**" and "**to manage**" as they appear in the Cambridge dictionary:

- management: the control and organization of something

- to manage: to be responsible for controlling and organizing someone or something

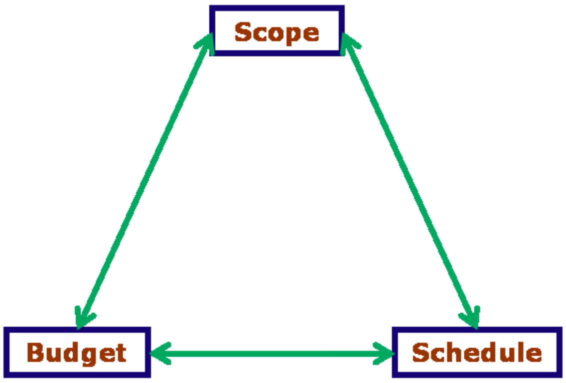
Here are definitions of "**project management**" (quoted from pmi.org and method123.com respectively):

- "Project management is the application of knowledge, skills, tools and techniques to project activities to meet project requirements."
- "Project management is the discipline of planning, organizing and managing resources to achieve the specific objectives of a project."

The above statements may be extended to "**program management**", which refers to the management of multiple interdependent projects.

Project management triangle

The key attributes of a project may be summarized in the so-called "Project management triangle", with "**Scope**" (or "Requirements"), "**Schedule**" (or "Time") and "**Budget**" (or "Cost") as vertices representing the **primary three constraints of a project**.

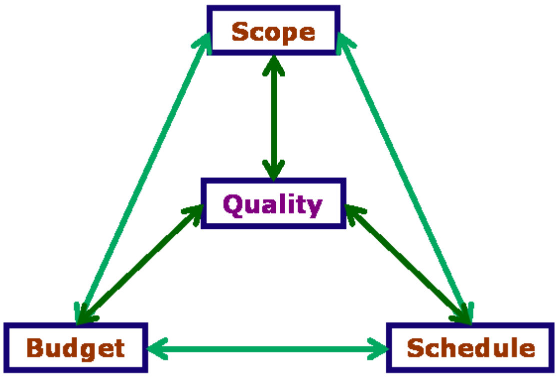


The **scope** of a project is the definition of what it includes. The scope is often expressed in terms of the project's "**deliverables**", ie its result or outcome (eg a product). It is generally useful to **also define** what is "**outside scope**", ie what will **not** be included in the project (although such a definition can obviously never be exhaustive…).

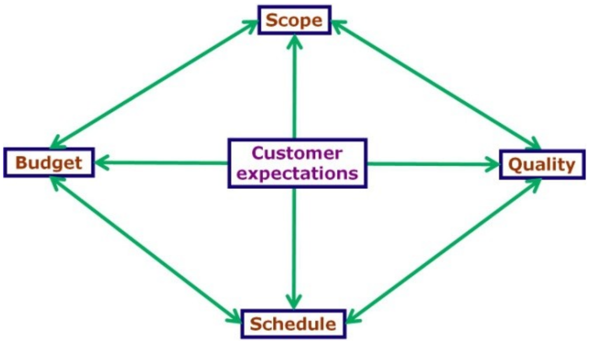
The **schedule** of a project refers to the lead time between the actual initiation of a project and its completion, its start and finish dates, as well as its intermediate milestones.

The **budget** of a project is the total amount of money which has been approved to be spent on the project in order to make it happen and to successfully complete it.

Quality, which is also a **major attribute and constraint** to be taken into consideration, is sometimes featured at the centre of the triangle. This attribute refers primarily to the quality of the project's result but it also relates to the quality of the processes involved in the project itself.



These four constraints may also be represented in a "**Project management diamond**", along with "**Customer expectations**" in the centre of the **lozenge**:



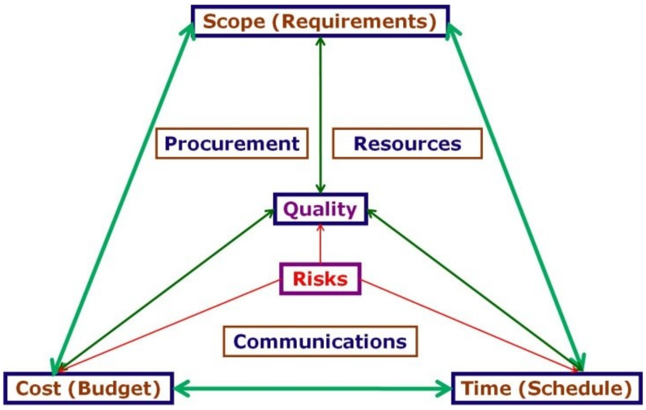
Customer expectations are the needs and wants of the "**project owner/sponsor**", ie the entity for which the project is undertaken, and of the "**end users**" of the result of the project, which is often a "product".

The message conveyed by these simple graphical representations is that **changing any one of the key attributes**, for example expanding the scope or tightening the schedule or reducing the budget, **will directly impact the other attributes**, for instance the quality of the resulting product, which may therefore not meet customer expectations. Such diagrams are helpful when discussing the issues of a project with its stakeholders…

Project management areas

- **The objective of project management is to ensure the completion of a project in compliance with the agreed scope, schedule, budget and quality requirements, in order to meet customer expectations.**

Meeting this objective requires the application of **knowledge, skills, tools, techniques and processes** in a number of areas summarized in the diagram below, which is an extension of the project management triangle.



Not all relationships between areas are featured in the diagram (in order to keep it readable…), but **direct relationships and interdependencies** nevertheless exist between virtually any two of the areas of project management.

The above diagram is useful as a graphical mnemonic of **the eight areas of project management** that need to be addressed in project planning as well as in project execution and supervision, as will be described at length further on in this guide.

The Project Management Institute (pmi.org) model features a **ninth area**, namely **Integration**, which includes the activities and processes required to identify, combine, unify and coordinate activities and processes in the other eight areas, which is what managing a project is all about!

2) The functions of a Project Manager ("PM")

General remarks

As stated previously, **a project needs to be managed** in order to produce the expected result. The person who manages a project is the Project Manager.

In this guide I use the abbreviation "**PM**" for "**Project Manager**", but "PM" may also stand for "Project Management".

Throughout this guide, although the PM may be male or female, I use the pronouns "he, him, his, himself" instead of the more politically correct but cumbersome "he/she, him/her, his/her, himself/herself"!
On some occasions I use the pronoun "you" to address you, dear reader, and of course the pronoun "I" refers to the author of this guide ("yours truly"). ☺

A project generally involves **two main parties**: the **project owner/sponsor** and the **entity in charge of project implementation** (or project execution).

The **project owner** is the "legal entity" or "natural person" at the origin of the project who specifies its requirements. The term "**project sponsor**", which specifically refers to the person or entity that funds a project, is often used as a synonym of "project owner", and in some cases the project owner and the project sponsor may be a single entity.

The **entity in charge of project implementation** (or project execution) is the "legal entity" or "natural person" that coordinates and supervises the implementation (or execution) of the project, in compliance with the requirements, the schedule and the budget that have been specified by the project owner/sponsor, who is often referred to as the "**client**". If the execution of the project is **outsourced**, the entity in charge of project implementation is usually called the "**main contractor**". The main contractor may use the services of "**subcontractors**" for the execution of parts of the work.

From the standpoint of a contractor, for example a software development company, an "**external project**" is a project owned by a different legal entity, for example a publishing company.

If the project owner and the entity in charge of project implementation belong to the same legal entity, the project may be viewed as an "**in-house project**" or "**internal project**", although the owner and the entity in charge of project implementation may belong to different parts of a particular company or organization.
For example, a company's top management or its Human Resources department or one of its Product divisions, as project owner/sponsor, may entrust the development of a business application or website or other product to the company's IT department, which will therefore play the role of entity in charge of project implementation.

The project owner usually appoints someone as the primary interface with the project manager appointed by the entity in charge of project implementation.

In many cases, the person assigned to that key role for the project on the owner side is a "**Business Analyst (BA)**" or "**Business Expert**". The BA has in-depth knowledge of the specific business operations to be impacted by the project; he represents the project owner's interests, as well as those of the future users of the product or service to be created; he is responsible for writing the requirements specification and for coordinating and supervising the project on the owner's behalf.

The BA may actually play the role of a **project manager on the project owner's side**, who may be called "**overall project manager**". In order to make the project a success, he needs to cooperate very closely and efficiently with the **project manager working for the entity in charge of project implementation**, who may be called "**implementation project manager**".

Such a situation is often encountered with projects for which the owner needs to outsource parts of a project to external companies, because the full range of competencies required to carry out the project is not available in-house. A frequent example of outsourcing is software development.

In certain situations, a PM may have **full project management responsibility, as owner as well as the person in charge of the execution of an internal project**, under the control of his management (the project sponsor).

As will be detailed further on in this guide, projects may be divided into "**subprojects**", each of which may be led by a "**subproject manager**" (or "**specialized project manager**").

> *In addition to my main role as Director of the Multimedia Reference Products division at Hachette, I was "Project Director" for the Hachette Multimedia Encyclopedia ("EHM"), with full responsibility for the project, under the supervision of two levels of management (the GM of the Encyclopedia & Dictionaries department and the CEO of Hachette Livre) and, of course, of a financial controller. My direct reports included an Editorial Director, with the role of editorial PM for the EHM, and a Data Engineering Director, with the role of technical PM for the EHM. The development of the EHM software, for most of its versions, was outsourced to an external company, which had naturally assigned a PM to the project.*
> *Multiple relationships were thus involved: between myself and the development PM, between the specialized Hachette PMs and myself, between the specialized Hachette PMs and the external development PM.*

Close cooperation between all PMs involved in a project is not always easy, but it is essential. Therefore someone has to make sure that cooperation is effective and that the project moves ahead smoothly towards successful completion. That is one of the major roles of the **overall project manager**, who should act as a true **project leader**.

> See the following site for **more information about Project Management and Business Analysis**:

>> https://pmhut.com/the-yin-and-yang-of-project-management-and-business-analysis

What does a project manager actually do?

This section describes the functions of a hypothetical **overall project manager** who has full responsibility for leading a project.

Metaphorically, the PM should "inhabit" the project, consider it to be his property, virtually on a 24/7 basis (to carry things to extremes). Conversely, the project inhabits the PM!

The PM is the master of the "**Project Office**" specific to his project. The Project Office is the main place of work where project management is performed; the project team members may however be distributed over several locations.

In large organizations there may be a "**Project Management Office (PMO)**", namely a department which provides project management standards, guidelines, advice, training, documentation, etc. to project managers throughout the organization.

The **PM's basic role and goal** (in his professional life) consists in making every effort, with the means at his disposal, to **achieve the objective** that has been set for a project, which may be summarized as follows:

- **successful completion of the project in compliance with its scope, schedule, budget and quality requirements, in order to meet customer expectations.**

By the way, any objective should be "**SMART**", ie:

- **Specific**: well defined, clear (as opposed to fuzzy).
- **Measurable**: tangible, with quantifiable results expected.
- **Attainable**: achievable, reasonable (given the constraints).
- **Realistic** (as opposed to utopian or wishful thinking).
- **Time-bound** (with at least a deadline for completion).

This is true for objectives set for a project or for a person (in particular the PM).

The **PM** is not necessarily the initiator of the project, but he **should preferably be involved at the earliest possible stage of the project and take full responsibility until its completion** (a change of PM in the course of a project is not desirable, but it may happen, and it is sometimes even necessary…).

The **PM** is not necessarily a product designer, but he **should actively participate in the product's functional design review**.

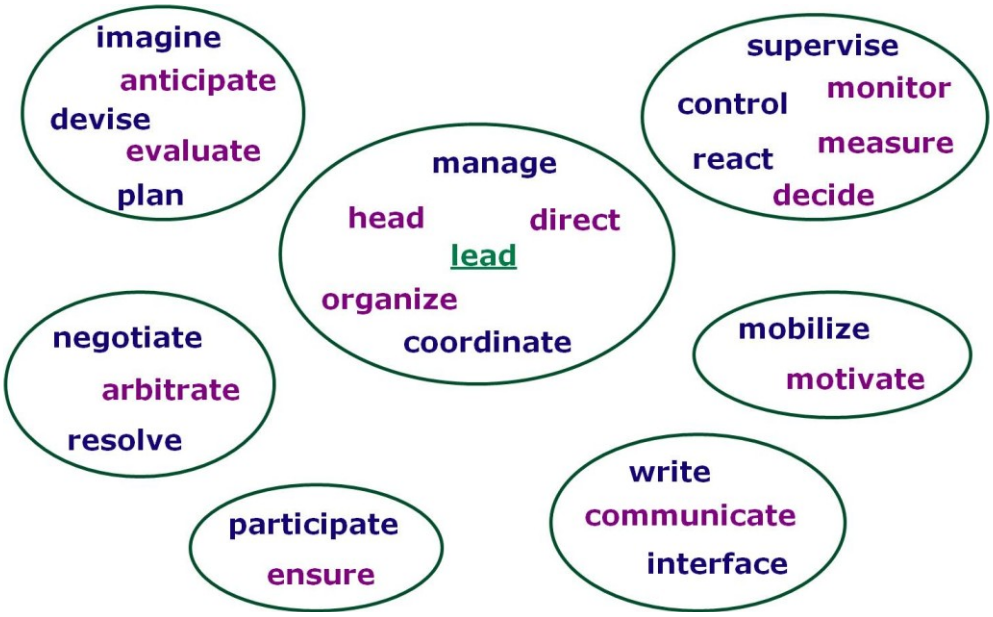
The **PM** is generally neither an editor, nor a developer nor a marketing expert, but he **should be familiar with the functions** of the various contributors to the project, and speak (or at least understand) their **specialist languages**!

A project may be compared to a work of music. The **role of the PM** is comparable to that of a **composer and conductor**: the PM writes the score and ensures that it is executed harmoniously by all the players. A good project manager is also a **true leader**.

Keywords

Here are a few **keywords that characterize the actions of a PM**: imagine, evaluate, anticipate, devise, plan, write, organize, coordinate, mobilize, motivate, supervise, monitor, control, measure, react, arbitrate, resolve, participate, ensure, negotiate, decide, interface, communicate, manage, direct, head, lead.

Those keywords are grouped by **skill sets** in the following diagram.



As a project manager, you will need to...

- **imagine** what the end result of the project will be and how to get there;
- **evaluate** the resources that are necessary for the execution of the project, for each of its phases;
- **plan** the project as a whole as well as each of its subprojects;
- **anticipate** risks, obstacles and problems that might occur, and **devise** appropriate contingency plans;
- **write** (or participate in writing) the requirements specification and **participate** in developing and/or reviewing the functional design specifications;
- **organize** the project, the project office, the project team, and divide up the work;
- **coordinate** all the players and their contributions;
- **mobilize** all contributors and **motivate** the project team members;
- **supervise** and **monitor** the work of all contributors;
- **control** and **measure** the progress of the various tasks and the progress of the project as a whole, as well as any possible deviation from the project's scope, schedule, budget and quality requirements;
- **react** as quickly as possible to any deviation (or other problem) by taking adequate corrective **action**;
- **arbitrate** disputes, **resolve** conflicts and **manage** the resolution of problems;
- actively **take part** in the testing of the product at each stage of its development, in order to **ensure** its quality, of which you, the PM, are the **guarantor**;
- **negotiate**, if the need arises, with the project owner/sponsor, with suppliers, service providers and contractors, as well as with your management and your financial controller;
- **make decisions** as required throughout the duration of the project, within the limits of your delegation of authority;
- **interface** between the project team and the "outside world";
- **communicate** information on a need-to-know basis to all parties concerned, at each stage of the project;
- **manage (direct, head, lead)** the project!

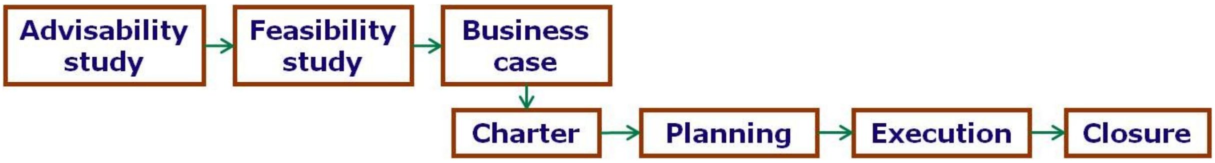
Note that the word "**lead**" is underlined above (and here!), because **leadership is an essential function of a PM**.

The various functions of a PM, which will appear in their full context in subsequent chapters of this guide, are performed throughout the life cycle of a project, an overview of which is provided in the following chapter.

3) The life cycle of a project

General remarks

This chapter provides an overview of the various phases in a project's life cycle, as shown in the following diagram. Details are provided in subsequent chapters of this guide.



Another high-level description of the sequence of phases involved in transforming an idea into a project then into a product is the following: **evaluation, initiation, preparation (planning & organization), product creation/development, closure**.

From idea to project

At the root of any project, there is an idea (or a concept). An idea may be interesting or crazy, workable or completely utopian, so it must be evaluated before turning it into a project or dropping it.

If an idea is retained, it is usually the PM's responsibility to turn it into a concrete project, regardless of whether the PM is the originator of the idea or not.

In a professional context, the idea may originate within a company (or organization) or it may come from an external party.

In order to **qualify as a potential project**, the idea should at least:

1) **correspond to a need** identified by the company, for its customers or for itself;
2) **be consistent with its strategy**;
3) **be workable** in principle;
4) **be economically viable** (unless the exception to this rule can be justified!).

Note that **innovative ideas** are not necessarily derived from needs expressed by potential users. Actually, such ideas often result in creating a need and, in some cases, a huge new market!

In order to determine whether undertaking a project is of any interest and whether the objectives of the project under consideration are achievable, an **advisability study** and a **feasibility study** need to be conducted.

Advisability study

To establish whether it is advisable to turn an idea into a project, the PM (or whoever is in charge of the study) needs to take into account information concerning the market, the competition, the company's needs and strategy, other projects that may be under way, etc.

In some cases, the advisability study may be very straightforward, since the justification for a project may simply be the absolute necessity to satisfy a need of the company or to solve a serious problem.

If the result of the advisability study is positive, the next step towards an actual project is a feasibility study.

Feasibility study

The feasibility study determines whether the objectives of the project are achievable, given editorial, technical, financial and timing constraints, as well as any possible risks.

In order to conduct the feasibility study, the PM (or whoever is in charge of the study) needs to imagine, at least as a broad outline, what the result of the project will be and how to achieve it, which requires a **summary project description** and a **summary project plan** including an estimate of the resources, the budget and the schedule involved.

The PM may require assistance from experts in order to conduct the feasibility study.

Business case

If it is believed, after an advisability study, a feasibility study and careful consideration, that the matter (the "idea") deserves to be taken further, a **business case** (or **business plan** or **project proposal**) must be presented to those of the **stakeholders** who have the **authority to decide** whether to go ahead with the project or not. The main stakeholder in this respect is of course the (potential) **project sponsor**, who, all being well, will **fund the project**.

The business case usually takes the form of an "**information package**" describing the **project**, its expected **result** and the **benefits** it will provide; it should also include a **risk assessment** and a **Profit & Loss ("P&L") evaluation**.

The project's estimated P&L is usually established by the PM, possibly with the help of a financial controller, and, in the case of a commercial product, in close cooperation with "Marketing & Sales", which is in the best position to provide product pricing estimates, sales forecasts and an evaluation of launch and other promotion costs.

The **basic purpose** of the business case is to provide **decision-makers** with the information they need to understand the project and to form an opinion of it.

- ➢ The "**Go / No go**" decision is usually based on the business case.

The business case may have been prepared independently of the PM, before he was appointed, although it is advisable to involve the PM (or "potential PM" at this stage of the "potential project") as early as possible in the process.

An experienced PM should neither have much difficulty nor require too much time to develop a business case, thanks to his knowledge of previous projects.
As mentioned above, the PM will however generally need to consult a few specialists, for example editors, developers, marketing and sales people, in order to validate the project's feasibility and its commercial (or other) interest.

> See the following article for **more information on the notion of "business case"**:

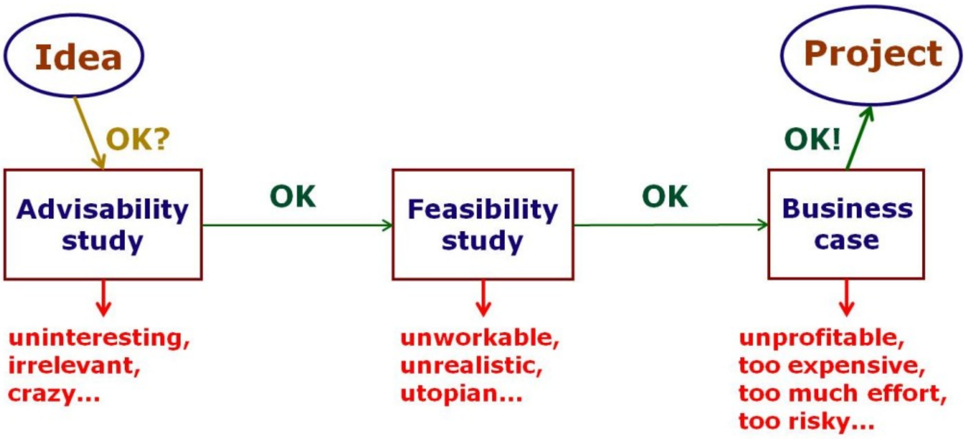
>> en.wikipedia.org/wiki/Business_case

I strongly recommend creating a short version of the business case, a so-called "**elevator pitch**" version that is liable to pass an "**elevator test**", ie which can be used **to "sell" the project in a matter of minutes**. An elevator pitch is particularly useful if lobbying is required to get the project accepted (with appropriate funding…).

> See the following article about **elevator pitches**:

>> elevatorpitchessentials.com/essays/ElevatorPitch.html

The following diagram provides a summary of the initial **steps leading from an idea to a project**.


Idea
OK?
Advisability study
OK
Feasibility study
OK
Business case
OK!
Project
uninteresting, irrelevant, crazy...
unworkable, unrealistic, utopian...
unprofitable, too expensive, too much effort, too risky...

A project that has been rejected may be given a **second chance**. In this case, the above process (advisability study, feasibility study, business case) should be repeated, at least partially, for a revised or alternative project.

Note that **alternatives**, each documented with their "pros and cons", may be featured in the initial business case in order to give decision-makers a broader choice.

Here are a few **examples of evaluation of ideas**:

> *In the early 1990s, at a time when CD-ROM technology was still very recent, Hachette decided to explore the possibilities of the technology for its line of business, namely content publishing. Hachette digitized one of its dictionaries and structured it in SGML (the ancestor of XML) in order to exploit it in "electronic" form on a CD-ROM, named Zyzomys, which was sold directly to schools and via retail outlets to the general public.*
>
> *Following this successful experience, other dictionaries were digitized and structured, and were used by companies such as Franklin and Sony, under licence from Hachette, on a variety of portable devices (Bookman, Data Discman). Concurrently with experimenting "electronic publishing", Hachette undertook the creation of an eighteen-volume encyclopedia in print form. Named Axis, it was put on the market in 1993 and sold directly to schools and families by a dedicated door-to-door sales force.*
>
> *A CD-ROM was added as an optional item to the Axis offering. The CD-ROM contained the text of the Axis encyclopedic dictionary along with thousands of multimedia assets (photos, drawings, animations, videos, sound…). It featured a very powerful advanced search tool which was especially appreciated in schools, in particular for its pedagogical virtues.*
>
> *In the same timeframe, the Hachette Multimedia Dictionary on CD-i (interactive CD), coproduced by Hachette and Philips, was released for the general public and "bundled" with Philips CD-i players, and the Hachette Multimedia Dictionary CD-ROM project was launched, based on the assumption that there would be greater demand for CD-ROMs than for CD-i products (which turned out to be true: the CD-i was discontinued in 1996).*
>
> *Over that period of five years or so, Hachette had acquired a solid experience in electronic publishing. It had also created a major encyclopedia, part of which was available on a multimedia CD-ROM.*

At that stage, Hachette was convinced it should become a major player in the emerging "multimedia publishing" market, in particular for "Reference" products, ie dictionaries and encyclopedias.

Furthermore, Hachette's American subsidiary, Grolier Publishing, was very successful with its "Grolier Multimedia Encyclopedia" CD-ROM, the first of its kind, which was available in English-speaking markets long before the arrival of Microsoft Encarta.

Finally, Hachette knew through competitive intelligence that Microsoft was preparing the French version of Encarta for release in 1996, that the development of a CD-ROM version of the Encyclopaedia Universalis (the French equivalent of Encyclopaedia Britannica) was about to be undertaken, and that one of its main competitors, Havas Interactive, was setting up a multimedia publishing organization.

So, in 1995, Hachette decided to create a division dedicated to multimedia reference products publishing (which I was hired to lead). Among other assignments, that division was given the responsibility of transforming the idea of a full-blown multimedia encyclopedia into a project, then of course into a commercial product (the "Encyclopédie Hachette Multimédia (EHM)").

Many ideas for products submitted to Marketing & Sales by the editorial teams at Hachette were rejected because the answer to the simple question "how many units could we sell?" was something like "probably no more than ten!".

One interesting idea that my team at Hachette came up with (before the team was dismantled…) was a website where the general public could subscribe to online reference and educational resources developed by several of the company's editorial departments. Although the project would have been feasible, it was rejected by management for three main reasons:

1) the implementation cost was considered to be too high;

2) there was an indisputable risk attached to the projected revenues (the issue being: "will the general public accept to pay for such resources?");

3) my team's charter did not include implementing and operating an online product distribution platform.

Project charter

Once the business case has been accepted, the project is **initiated** and moves on to its **preparation** phase. One of the first tasks in this phase consists in establishing the **project charter**, which is a reference document that provides a **detailed description of the project**, as well as of its outcome (in a **summary requirements specification**).

The following terms may be considered as **synonyms of "project charter"**: "project mandate", "project definition", "project initiation document (PID)", "project overview statement (POS)" and "terms of reference (TOR) of the project".

The project charter should clearly document the **project's scope, schedule and budget** (however imprecise they may be at this stage of the project**)**. It should also define the **scope of the PM's responsibilities** and his **level of authority**.

This subject is covered in more detail in chapter 7.

Requirements specification and project planning

Expanding on the project charter, the PM must describe and plan the project very precisely, with a **thorough requirements specification** and a **detailed and exhaustive project plan**, which may be considered as a "**roadmap**", covering all areas of project management (as described in chapter 1).

These subjects are covered in more detail in chapters 8 and 9.

Forming the project team, choosing contractors

Before going ahead with the actual execution of the project, the PM usually needs to form a project team and to establish an **adequate organization**. In some cases, the PM will need to **recruit staff** and to **select contractors** as well as suppliers and service providers.

The choice of contractors generally involves evaluating proposals and estimates made by candidates on the basis of specific "**Statements of Work (SOWs)**" adapted to each type of work. SOWs, which are usually derived from the overall requirements specification, must be **precise and exhaustive**. Ideally, the PM should have specialists at his disposal to help write the SOW documents.

These subjects are covered in more detail in chapters 10 and 11.

Project execution

The implementation phase of a project involves execution of the tasks documented in the project plan.

Software development life cycle models and methodologies

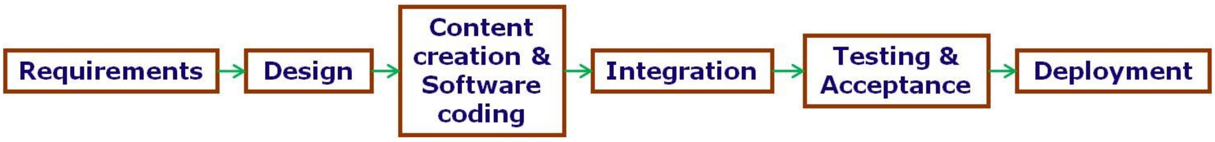
There are many models that describe the sequence of phases specific to a software development project: requirements, design, implementation (or coding), integration, testing, deployment (or delivery or entry into operation or launch).

Each particular model emphasizes certain features of a project's organization and dynamics. The choice of a model has an influence on the detailed definition of a project's execution phases, so, ideally, **the model should be chosen before planning the project**.

The following models are addressed in chapter 12 of this guide: Waterfall model, Incremental model, V-shaped model, Spiral model, as well as the Agile Software Development methodology and one of its applications, eXtreme Programming.

Major phases of product creation

The following diagram provides a summary view of the **major "work packages"** (sets of tasks) involved in the execution phase of a project that results in a product featuring content. They can be considered as **phases of product creation**.



Requirements

Although a requirements specification needs to be provided before a project can be planned, one of the first work packages involved in product creation is "Requirements" (as featured in the above diagram), because the requirements specification provided by the project owner is the **foundation for product design** and **may need to be reviewed and modified** before actually doing the design and/or as progress is made on designing the product.

Design specifications

Most projects involve work that requires very **detailed design specifications**. This is particularly true for software and website development, as well as for complex editorial projects.
Product **design specifications need to be written before the product is actually built**. The specifications are the "**blueprint**" for building the product.
Such specifications are usually devised and written in close cooperation with specialists in each area of interest, for example the editorial manager for editorial specifications, the technical manager or software developer(s) for functional and technical specifications.
Detailed technical specifications are generally written by the developer(s) and/or a development PM. Functional design specifications need to be **reviewed and validated by the overall PM** (possibly with the help of appropriate specialists in the project team).

This subject is covered in more detail in chapter 13.

Content creation, software coding, integration

Content creation, coding (implementation) of the software (which is the "container" part of the product) **and integration** (of content with the container) are the work packages corresponding to the actual building of the product.

Content creation and software coding can generally be performed **in parallel** (or at least with a certain degree of parallelism).

These subjects are covered in more detail in chapters 14 and 15.

Testing and acceptance

The implementation of a project materializes in the form of "**deliverables**", which are generally successive, more and more complete versions of a product, corresponding to various stages of content and software integration, as well as documentation, etc.

It is the PM's responsibility or that of a subproject manager to organize **testing** of deliverables beyond the basic technical tests that are usually carried out by the developers themselves under the supervision of a development subproject manager.

The PM should take an active part in **quality control**, which involves testing, since he is ultimately responsible for the quality of the finished product and **accepting** it before deployment/launch.

This subject is covered in more detail in chapter 16.

Product deployment/launch

The deployment/launch of the resulting product must be prepared in **close cooperation with Marketing & Sales** (particularly in the case of a commercial product) or with any other entity that will be in charge of the product's operation.
The **PM**, as the primary interface between the project team and the "outside world", is the **official source of project and product information** for Marketing & Sales (which usually includes Customer Services / Technical Support).

The PM provides the product description and often helps Marketing & Sales **translate product features into user benefits**, devise a sales pitch, brief a communications agency and advertising people, write a press release, etc. Marketing may also ask the PM to take part in press conferences, in product demonstrations, in interviews with journalists, etc.

This subject is covered in more detail in chapter 18.

Project direction/supervision, monitoring and control

The PM **directs** project execution, which involves **coordinating** and **supervising** all of the tasks (activities) that are involved in the **execution of the project** and the **creation of the resulting product**. For that purpose, the PM must keep himself **informed** about the progress of the tasks; he needs to **identify any deviation** from the requirements, the schedule, the budget, etc., and, whenever necessary, he must **take corrective action** as soon as possible.

The PM needs to closely **monitor** and **control** the work of all contributors to the project, which generally involves using a "**project dashboard**" consisting of various tables, charts, diagrams and other documents that represent "**performance indicators**", with up-to-date information on the project's **progress** (this information may also be used for reporting purposes). Regular progress and coordination meetings enable the PM to take stock of the status of the project, to draw conclusions and to make appropriate decisions.

This subject is covered in more detail in chapter 17.

Project closure

The end of a project usually coincides with the launch of the product which has been gestating since the initiation of the project.

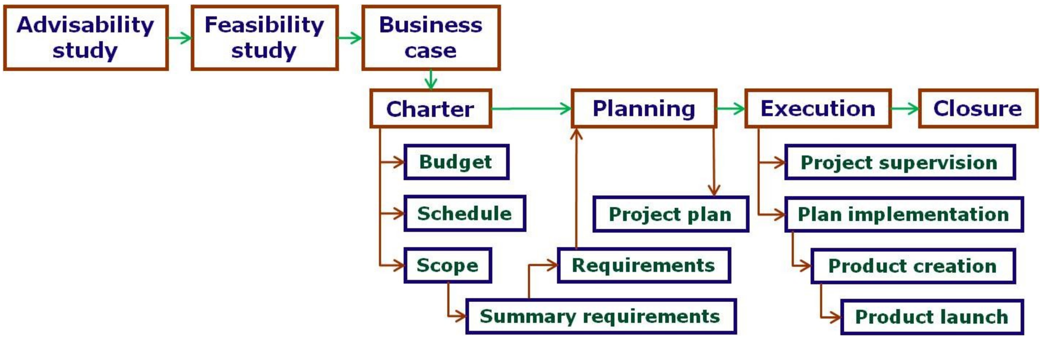
Project closure involves **releasing resources** that are no longer needed, **terminating contracts** and making sure **adequate project documentation** is available for follow-up purposes and future reference.

Before a project is definitively closed, the PM should submit it to a **post-mortem analysis**, in order to learn lessons for future projects. Ideally, the results of this analysis should be fed into a "**knowledge base**".

That exercise should take into account any available **user feedback** on the product. **Users** are indeed a **very valuable source of information** for future projects such as the development of new versions of the product and/or its maintenance.

Summary diagrams

Here is a diagram showing the key phases in a project's life cycle, including more details than those provided by the diagram featured at the beginning of this chapter:

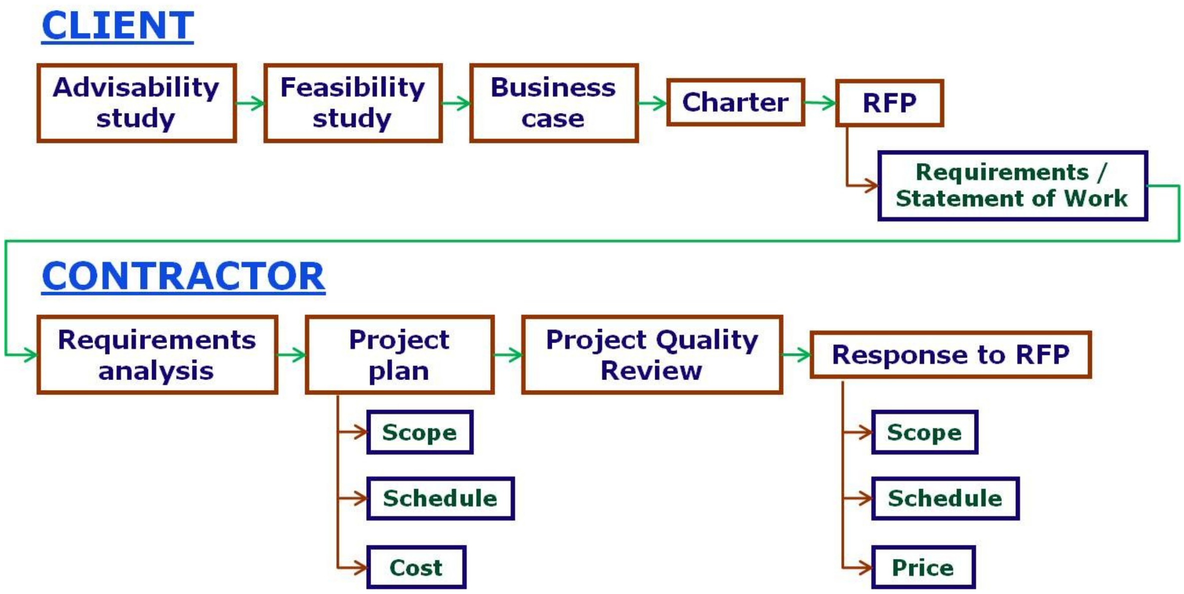


From a **contractor's viewpoint**, a project is generally **a service to be provided to a client** (which may be internal or external), and, as far as the contractor is concerned, the client's "business case" is generally a given. The project nevertheless has to make sense and be economically viable for the contractor, so there is a need for a **business case on the contractor's side**, which involves preparing a project plan that meets the potential client's requirements while being acceptable to the contractor, in particular from a financial standpoint.

The presentation of the business case to the contractor's management may be done on the occasion of a so-called "**Project Quality Review**", following which the "go ahead" for the project may or may not be given.

Such a review is generally conducted by a contractor before responding to a "**Request for Proposal (RFP)**", and it may very well result in a "no go" decision, ie the contractor may decide to not respond to the RFP.

The following summary diagram shows the steps leading from an idea to an **RFP** on a client's side, as well as the steps taken by a potential contractor towards **responding to the RFP**, ie submitting a proposal to the (potential) client.



Note in the above diagram that the contractor's response to the RFP features a "**price**", as opposed to a "cost". Indeed, not only should the price paid by the client **cover the cost** of the work that has been estimated by the contractor as part of the planning process but it should also **include a profit margin** in order for the project to be **economically viable** from the contractor's standpoint.

This subject (cost vs price) is covered in more detail in chapters 6 and 9.

4) The life cycle of a product

Product deployment/launch

One of the important stages of **the final phase of a project is the preparation of the deployment/launch of the resulting product**. To put it simply, the end of life of a project marks the beginning of the life of the resulting product, ie its **entry into operation** (which may be commercial or not).
In actual fact, the product is in gestation throughout the life of the project.

As mentioned previously, in the case of a commercial product, the PM is often involved in helping Marketing & Sales with the product launch preparation and sometimes the launch itself.

Product evolution

The first version of a product that is supposed to evolve over time results from the "**initial project**" (or "base project") and each of the following versions generally requires a specific new "**follow-on project**".
This is generally the case for a "massive" change or enhancement of a product's content, as well as for new releases of the software (corrective releases, update releases, major revisions…).

If it is possible to imagine from the outset what the **successive versions** of the product will be, then their features and constraints should be **integrated into the initial project**, in order to make the execution of corresponding follow-on projects easier, to increase their chances of success and to minimize their cost.

Products, in particular software, generally require **maintenance**, which may be provided on an ongoing or occasional basis, as specified in a maintenance contract if software development is subcontracted by the project owner. Depending on the context, the cost of maintenance may be explicitly charged by the contractor to its client or included in the overall "price" of the project invoiced to the client.
In some cases, product maintenance is treated as a follow-on project.

A version of a product may be distinguishable from the previous version by nothing more than an **evolution of its content**, with no change to the software. In this case, the making of a new version of the product should not require a new development project but the application of a **process** that has been specified and implemented as part of the initial project.

> *For the Hachette Multimedia Encyclopedia ("EHM") on CD-ROM (and DVD-ROM), it had been decided that users would be given the option of downloading content updates on a monthly basis. This function was featured in the specifications, so the developers devised a mechanism for updating the EHM content, initially delivered on the CD-ROM, with data downloaded to the user's hard disk from a dedicated website. The software process for content update was thus automatic.*
> *That process was also applied for updating the EHM software itself (for minor bug fixes).*
>
> *The latest version of the online EHM was complemented with a sophisticated "back office", including a chain of software applications to extract new data (additions or updates) from the (Oracle) database, and to index, reformat and upload the data to the EHM website.*
> *Thanks to that almost fully automatic process, which had been specified as part of the initial project, editors were able to trigger an update of the online EHM, usually on a weekly basis, without requiring any intervention from the developers.*

As a general rule, and in particular for a website, the initial project should include the design and development of a set of administration and editing tools, sometimes called "**back office**", which will in particular enable the people in charge of the product's content (editors, etc.) to update it without having to bring in technical specialists.

Specifications of the back office need to be prepared in close cooperation with its future users so that the tools that are developed perfectly meet the editors' needs and make the tasks of content addition and updating simple and efficient.

Marketing & Sales sometimes require **demonstration versions or "light" versions** (reduced versions) of a product.
If this requirement is integrated into the initial project, additional projects corresponding to these "**by-products**" will generally be simple, short and inexpensive.

> *With promotional operations in mind, Hachette Multimedia's Marketing had asked for special versions of the EHM featuring content limited to one or several topics (Art, History, Science and Technology, etc.) and software limited to a subset of the product's functionalities.*
> *That specific constraint had been specified in the initial project and was taken into account by the EHM software developers. As a result, each of the required EHM by-products could be made at a very low cost within just a few days.*

End of life of a product

All good things generally come to an end, even products!

The end of a product is an integral part of its life cycle and should be taken into account in the corresponding project.

The end of a product is characterized by the fact that it becomes "**frozen**": its content and its software will no longer evolve. However, this does not necessarily imply that the commercial operation of the product needs to be stopped immediately.

> *The Hachette Multimedia Atlas (AHM) and the Hachette Multimedia Dictionary (DHM) on CD-ROM were frozen in 2001 and 2002 respectively on the occasion of the production of the AHM's 6th version and the DHM's 8th version.*
> *Marketing had however decided that these products would stay in Hachette Multimedia's catalog for as long as they could be sold. The products were repackaged, without any reference to a specific year of release, and their recommended retail price was progressively dropped to the equivalent of 19.90 euros.*
> *The AHM and the DHM thus remained on the market for many years after the effective end of the product (from its producer's perspective). It even turned out necessary to manufacture several hundreds of AHM CD-ROMs in the first half of 2007 in order to meet customer demand!*
>
> *The final version of the Hachette Multimedia Encyclopedia (EHM) on CD/DVD-ROM, the EHM 2007, was produced in June 2006. In order to extend the product's life as long as possible, its software was made compatible with the new Macintosh computers featuring Intel Core Duo processors.*
> *The EHM 2007 remained in its distributors' catalogs for as long as there was demand, and as long as salespeople did not consider the product to be totally obsolete.*
> *Marketing did not want to remove the mention of the date ("2007") from the EHM box (as I had recommended…), so its obsolescence became apparent as soon as the "2008" versions of its competitors (the Larousse Multimedia Encyclopedia and Microsoft Encarta) were released on the market.*
> *Despite this major disadvantage, the EHM 2007 could still be found on retail store shelves in 2008 and could still be purchased online throughout 2009…*

The **decision to freeze a product** may be made for various reasons, for example: its maintenance cost is judged to be too high; its evolution is made impossible because its original developers are no longer available, and having its software taken over by new developers would be too risky and too expensive; the decision to produce a brand-new replacement product has been made.

Once a software application is frozen, any **technical incompatibility** with its environment (which never ceases to evolve…) leads to problems for users and therefore for whomever is in charge of the frozen product (its original project owner, its product manager, its publisher…).

It is essential to figure out and plan how to deal with such problems. The PM is often involved in devising solutions, in close cooperation with the appropriate stakeholders, eg Marketing & Sales, in particular the product manager and Customer Services.

If another product can be substituted for the faulty product, the solution is simple, though it may be costly (eg for the manufacturing and shipment of the replacement product).
A more cost-effective solution that may be satisfactory for a large proportion of users consists in proposing an online product as a **replacement** for an offline product that has become unusable.

> *There was a network version of the EHM CD/DVD-ROM that encountered technical problems in a growing number of configurations. The cost of solving the problems being considered too high with respect to the volume of sales of the product, I decided (with the approval of Marketing & Sales) to discontinue the network version of the EHM, following its 9th version dated "2006".*
> *The few customers (less than a dozen) who had reported major problems with the network version of the EHM 2005 and 2006 were satisfied with being provided, free of charge, with a stand-alone EHM 2007 DVD-ROM that could be fully installed on each of the user workstations, or with a free subscription to the online EHM (which had the advantage of being frequently updated).*

If there is no available replacement product, customer complaints may require **refunding** the faulty product at its purchase cost.

It is wise to document in a very explicit manner on a product's packaging the **configuration required** for a product to work properly. Doing that may avoid having to refund a product that does not work with a configuration not included in the description on the box.

> *Some Hachette Multimedia products worked properly with Mac OS 9 but not with Mac OS X. The explicit warning "Does not work with Mac OS X", which I convinced Marketing to print on the boxes, avoided refunds (but not necessarily discontent of customers who had not paid enough attention to the configuration section on the back of the packaging…).*

If the **termination** of a subscription-based online service is considered, it is essential to set the date as of which subscriptions (or renewals) will no longer be accepted.
Furthermore, for the sake of rigour and honesty, the service should remain available to customers until the final date of validity of the last registered subscription.
If it is impossible to maintain the service until the above-mentioned date, it will be necessary to refund subscriptions in proportion to the duration of unavailable service.

> *The content of the online EHM was frozen at the end of November 2007 (its software was frozen in June 2006). Having decided to terminate the online EHM service on September 30, 2009, Hachette stopped accepting subscriptions to this service on October 1, 2008, and started proposing subscriptions to the online Larousse Encyclopedia instead of the online EHM. (Note that Larousse belongs to the Hachette Group.)*

Factors influencing the life of a product

There are many factors that influence the life of a product.
One of them may be the absolute necessity, due to the nature of the product, to **enhance and update its content**, as mentioned above. Most of the other factors impact not only the content of a product but also its software. Such factors are addressed below.

Market and competition

The market and competition must obviously be taken into account in order to ensure that a product remains adapted to market requirements as well as competitive.

Keeping up-to-date on the evolution of the market and being knowledgeable with the **products of competitors** is necessary not only for a product manager but also for a PM.
As a PM, you should allocate part of your project budget to purchasing competitors' products or subscribing to them, and you should get to know them in depth by actually using them.
Indeed, you must be able to position the product "you" are creating with respect to its competitors, in terms of content, functionality and performance, in order to draw conclusions regarding the evolution of the product.
Precious information about competition may be obtained from **commercial partners**, with a little help from Marketing & Sales if necessary.

The market often demands **novelty** and even **innovation**.
Marketing & Sales people, in particular those working for distributors and retailers, want clear and simple **sales arguments** to compare a product with its competitors, and they are generally fond of products with **distinctive and "exciting" new features** that can be easily explained to potential buyers in terms of **user benefits**.

> *For example, here are some innovative (or at least new) features of the EHM that were highlighted as sales arguments and which marked the life of the product:*
>
> - *full hypertext (first version of the EHM),*
> - *doubling the number of multimedia assets (EHM 2000),*
> - *3D rendering of historic monuments and sites (EHM 2000),*
> - *natural-language queries (EHM 2000),*
> - *summary articles for schoolchildren (EHM 2000, more added in subsequent versions),*
> - *"panorama" of search results comparable to the front page of a newspaper (EHM 2005),*
> - *compatibility with Windows XP, Mac OS X and Linux (EHM 2005).*

User requirements

User requirements generally evolve as users gain experience with products and therefore become more knowledgeable and more demanding. In that respect, users should be provided with a **channel to communicate** their remarks and requests in the most direct possible fashion. One or several persons need to be assigned to dealing with customer feedback, and they should be required to forward relevant information to the PM so that it may be taken into account for future versions of the product.

> *As regards Hachette Multimedia ("HM") products, the hotline's phone number appeared on all boxes, in all manuals and on the HM website, where a full contact list (with e-mail addresses) was also provided to let users know whom to contact depending on the type of request. Furthermore, two specific addresses appeared on the EHM website, one for editorial remarks, the other for technical issues.*
> *Finally, analyzing searches made by users in the encyclopedia itself gave an indication of their areas of interest, which was one of the sources of information used to set the direction or priorities of the editorial team's work.*

Useful remarks and suggestions concerning products may also be made by **commercial partners** (sales reps, distributors, resellers, etc.) and of course **Customer Services**. Such feedback usually reaches the **Marketing & Sales** department, which is another good reason why the PM should entertain a close relationship with Marketing & Sales throughout the life of the product. Furthermore, the PM should not hesitate to go "fishing for information".

> *Some retailers, in particular large distribution chains, informed Hachette Multimedia of their desire to have an "entry-level" version of the EHM, at a relatively low price, given the profile of the majority of their customers. To meet that requirement, the project team designed, planned and produced what was named the "standard" version of the EHM, downsized to a single CD-ROM with half the number of media and less functionality than the "complete" DVD-ROM version.*

Price

The price of a product is generally influenced by market **demand & supply**.

> *In less than eight years, the retail price of the "bottom-of-the-range" version of the EHM on CD-ROM dropped from 100 euros to 30 euros, mainly due to competition.*
>
> *A particular iPhone application ("C"), which its developers did not intend to "give away", was in competition with two other applications: one ("B"), which provided almost the same content and functionality as C, was priced at 0.79 euro (with a "light" version free of charge); the other ("A"), which was a superset of C and B, was priced at 15.99 euros.*
> *In order to justify a price for C in the range of 2.99 to 4.99 euros, the developers imagined and planned to develop unique features (and sales arguments) that would make C a better, easier-to-use and more useful product than B and the subset of A.*

Price evolution often has a direct impact on a product. As mentioned above, it may be necessary to define a low-price, entry-level product. A high-price version of a product may nevertheless be maintained in the product range by providing more content and functionality than, for example, the previous year's version.

> *For several years in a row, the range of EHM CD/DVD-ROMs consisted of a "standard" edition on a single CD-ROM at 30 euros, a "complete" edition on two CD-ROMs or a DVD-ROM at 60 euros, a "deluxe" edition on a DVD-ROM at 100 euros, including the Hachette-Oxford English-French Dictionary as a substantial "bonus", and a "network" version, for which the price of the site licence depended on the number of users.*

Business model

More generally, the evolution of a product is often tied to the evolution of its business model, ie everything that relates to **sales channels and how revenue is generated**.

For example, the services rendered to users by a website may be subject to a fee or they may be free of charge. Some sites offer a basic service free of charge and a "premium" service requiring payment of a fee. The definition and extent of each set of services may evolve over time: new services may need to be added in order to remain competitive; the payment system may change; advertising space may be increased, etc.

> *The business model of the online version of the EHM went through many changes, most of which, summarized below, had an impact on the product and therefore required some amount of additional development effort:*
>
> - *EHM free of charge, with ads, on Club Internet, Wanadoo, Voila (2000), Yahoo! (2001),*
> - *EHM reserved to subscribers on AOL (2001),*

- *fee-based EHM "Pro", without ads, for the institutional market (2002),*

- *switch to fee-based model, with subset of content free of charge, on Wanadoo and Voila (2003),*

- *fee-based EHM, without ads and with subset of content free of charge, via the Hachette Multimedia website for the general public, and via the "Digital Kiosk for Education" for schools and libraries (2004),*

- *change of payment system for subscriptions (IDECOD → P-PASS, 2004),*

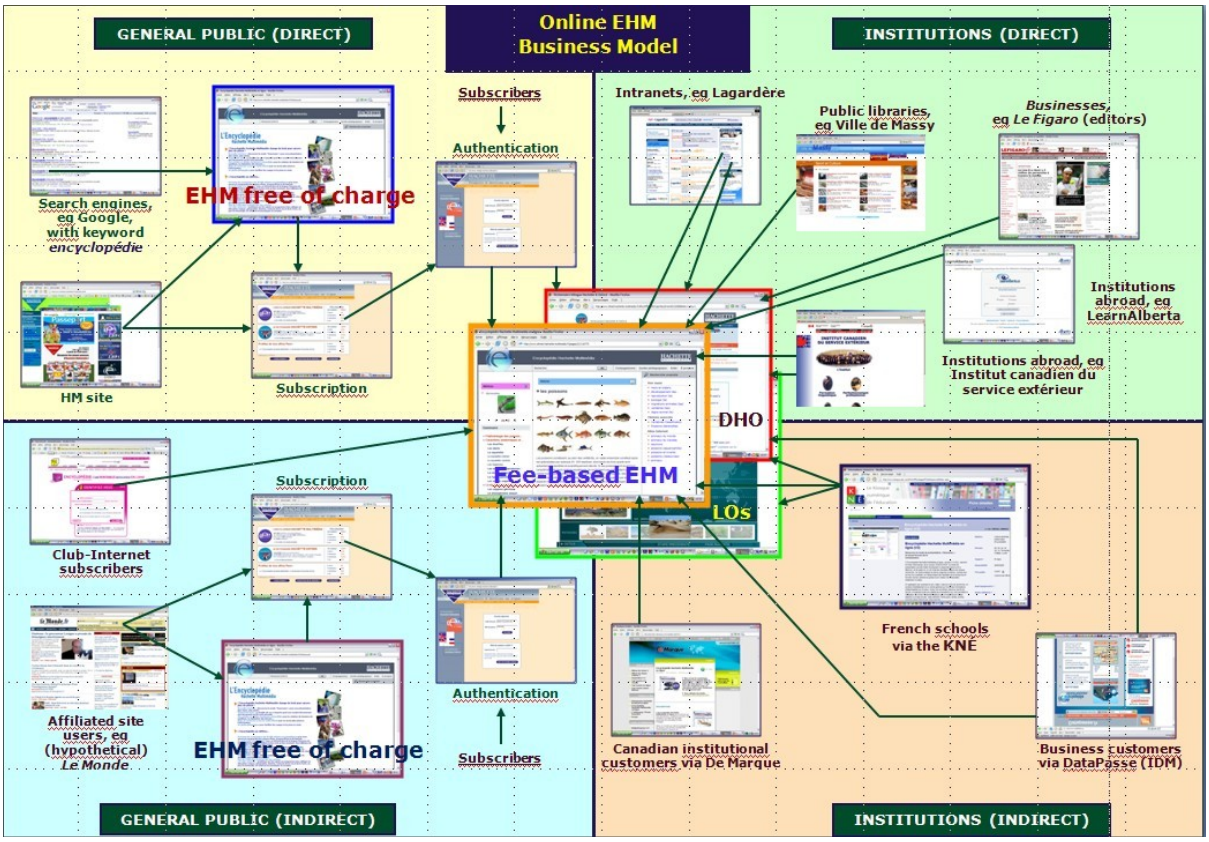
- *new fee-based EHM, without ads and with a subset of content and features free of charge, derived from the fully redesigned and redeveloped EHM on DVD-ROM (2005).*

As an example, the following diagram represents the business model of the online version of the EHM as it stood in the 2005-2006 period.

The implementation of the model required designing the online encyclopedia software in such a way that a small but consistent part of the encyclopedia's content could be made available free of charge, while the full content was reserved for subscribers.

It also required developing a subscription mechanism and a user authentication system, with a database of user information and administration tools, as well as plugging in a third-party payment system.

Finally, it required a "single sign on" (SSO) interface for user authentication via partner sites through which the online encyclopedia was made available.


GENERAL PUBLIC (DIRECT)
Online EHM Business Model
INSTITUTIONS (DIRECT)
Subscribers
Authentication
Intranets, eg Lagardère
Public libraries, eg Ville de Massy
Businesses, eg Le Figaro (editors)
Search engines, eg Google, with keyword encyclopédie
EHM free of charge
HM site
Subscription
Institutions abroad, eg LearnAlberta
Institutions abroad, eg Institut canadien du service extérieur
DHO
Fee-based EHM
LOs
Club-Internet subscribers
Subscription
French schools via the KNE
Affiliated site users, eg (hypothetical) Le Monde
EHM free of charge
Authentication
Subscribers
Canadian institutional customers via De Marque
Business customers via DataPasse (IDM)
GENERAL PUBLIC (INDIRECT)
INSTITUTIONS (INDIRECT)

If it is feasible, **foreseeable evolutions** of a product's business model should be taken into account during the **product design** phase, in order to make required changes as easy and inexpensive as possible.

Target platforms / Technical environment

A software product is designed to operate on "**target platforms**" consisting of **hardware** devices (computers, tablets, smartphones, e-readers…), **operating systems** (Windows, Mac OS, Linux, Chrome OS, iOS, Android, Windows Mobile…), **browsers** (Internet Explorer, Edge, Firefox, Safari, Chrome, Opera…) and "**plug-ins**" (QuickTime, RealPlayer, FlashPlayer…).

The **good working order** of a product is therefore dependent on its **compatibility** with such platforms, which make up what is also called the product's "**technical environment**".

The more **complex** the environment is and the more **dependencies** it creates, the higher is the **risk** of some form of **incompatibility** appearing at some (often unpredictable) stage.

The **choice of target platforms** is generally the result of a compromise. It is obviously desirable to aim at the broadest possible market, but it is also necessary to take into account the product's development, testing and maintenance costs, which depend on the multiplicity and complexity of the platforms (such costs are not always predictable…).

> *Unlike its competitors (encyclopedias from Microsoft and Larousse), all versions of the EHM on CD/DVD-ROM were compatible with Windows and Mac OS, and also, as of version 8, with Linux. This wide range of target operating systems enabled Hachette to benefit from the significant Macintosh user base in France, and as far as Linux is concerned, to communicate the fact that the EHM was the only encyclopedia on CD/DVD-ROM available in that more and more "popular" environment (praised by "techies" and institutions in particular).*
> *In 2007, the EHM's compatibility with Linux actually enabled Hachette to win a call for tenders issued by the local administration of the Bouches-du-Rhône (Marseille and region) for a Linux-compatible encyclopedia on DVD-ROM (although the 120,000 portable PCs purchased from HP were eventually configured with Windows!).*
>
> *The counterpart of choosing multiple target operating systems was significantly higher development and testing costs than those that would have been associated with a single target platform.*
> *In the case of the Macintosh environment, substantial software changes were required throughout the life of the EHM, in particular to make it compatible with Mac OS X then with the new Intel Core Duo processors that Apple had chosen as a replacement for PowerPC processors.*

Furthermore, writing software for multiple platforms often implies using **development tools** that are not necessarily optimized for all environments. Generally, one **environment** needs to be **favoured** to the detriment of the others, which should be done with full knowledge of the consequences.

> *A complete redesign and redevelopment of the EHM was undertaken for its third edition (dated "2000"). Because of a relatively low development budget, Java was chosen as the development language because of its multi-platform capability, with a "Java virtual machine" for Windows (developed by Sun Microsystems) and another for Mac OS 9 (developed by Apple).*
> *The EHM software thus became extremely dependent on external software.*
> *A major incompatibility appeared with Pentium 4 PCs, which was fortunately fixed for the Windows environment thanks to a new virtual machine provided by Sun, but a new release of the EHM had to be produced.*
> *In the Macintosh environment, the EHM encountered a series of technical problems, because the virtual machine from Apple turned out to be slower, less stable and more defective than that provided by Sun.*

In order to develop a reliable and efficient product, with a satisfactory level of performance, it is often advisable to **focus coding, testing and maintenance efforts on a single platform**, and to **minimize external dependencies**, thus avoiding the risk of higher-than-expected costs and lower-than-expected technical quality.

> *Choosing Java for the EHM 2000 project was probably a mistake. Furthermore, although it cannot be demonstrated, EHM sales might not have been substantially lower with a single target platform (PC/Windows), but it is certain that production and maintenance costs would have been much lower.*

The above-mentioned potential problems are certainly less important and less critical with websites. Indeed, **website compatibility problems** are generally limited to browsers and plug-ins, which are subject to frequent updates that users can easily download and install.
Furthermore, fixing a software bug on a website and updating its content are done much faster and more easily than, for example, producing a new release of a CD/DVD-ROM.

However, a web application may not necessarily work perfectly with all browsers. It should be **optimized and thoroughly tested for a choice of browsers** (not forgetting those provided on smartphones and tablets…), to the detriment of others, and users should be informed of recommended configurations.

5) Describing products – The 4 Ps

A convenient and methodical way of describing a product consists in detailing its "**Marketing mix**", divided into four sections corresponding to the **four basic components of a marketing plan or strategy**: the "**4 Ps**", namely:

- the **P**roduct itself (with its content, functions, features and other characteristics),
- its **P**rice (and pricing strategy and business model),
- **P**lace (ie where and how the product can be purchased or accessed),
- **P**romotion (ie the various activities undertaken to promote the product).

Product

A summary presentation of the product should describe its **content and main functions** in simple terms, emphasizing **benefits for the users** as well as **distinctive features** with respect to the product's possible competitors. A more detailed description of the product should be provided in its requirements specification.
In addition, the following characteristics of the product should be addressed.

Target users (or user groups), for example:

- general public, possibly restricted to an age group (eg 9-15) or to another type of segment (eg French-speaking persons interested in Business English),
- businesses and/or institutions,
- schools.

Target platforms, for example:

- Windows, Linux, Mac OS X, Chrome OS,
- Windows Mobile, iOS, Android, e-reader,
- Internet Explorer, Edge, Firefox, Safari, Chrome, Opera.

Delivery media, for example:

- CD/DVD-ROM,
- website (download),
- website (online service).

Price

It is necessary to have a rough estimate of the sales price of the product in its various forms early on in the project, in order to establish its **forecasted P&L** (profit & loss statement) and to position the product with respect to its competition. More generally, this component of the marketing mix covers the **pricing strategy**, which is related to the distribution channels ("Place") and the product's **business model** (ie how revenue, if any, is generated).
Note that a sales price may actually correspond to a subscription fee or a licence fee.

In the case of websites intended to be completely free of charge, there is no price attached to the product but if its business model features advertising revenues, such revenues need to be included in the P&L.

Place

The word "place" refers to "**where the product can be found**" and relates to distribution channels/processes, which may have an impact on the product requirements. Distribution costs must be featured in the P&L.

Here are a few examples of distribution channels/processes:

- indirect sales via wholesalers and/or retailers,
- indirect "OEM" ("bundle") sales,
- direct sales to the general public,
- direct sales to businesses and institutions,
- online sales with product shipment,
- online sales with product download,
- direct online subscription,
- indirect online subscription (via commercial partners),
- online service free of charge.

Promotion

This component of the marketing mix covers everything that is to be done to promote the product, for its launch and beyond. The corresponding **costs** need to be **included in the P&L**.

Here are a few examples of promotion methods and activities:

- evangelism,
- press releases,
- press conferences,
- sample products (free of charge) for journalists and "prescribers/evangelists",
- demonstration version of the product,
- leaflet and/or brochure,
- mailing/e-mailing,
- website, blog, Twitter and, more generally, viral marketing,
- website referencing and search-engine optimization (SEO),
- partnerships (eg sponsoring…),
- exhibitions (booth, demos, "goodies", etc.),
- advertising (newspapers, magazines, radio, television, cinema, Internet).

The promotional activities, as well as the other components of the marketing mix, are generally described in much more detail in a document prepared by Marketing, namely the product's **Marketing plan**, for which input from the PM is usually required (and no doubt useful…).

Note that the expression "**4 Ps**" is also sometimes used in project management to refer to four key areas, namely: **P**eople, **P**roduct, **P**rocess, **P**roject (or **P**olicies, **P**rocedures, **P**eople, **P**lant).

> See the following sites for **more information on the Marketing mix**:

>> netmba.com/marketing/mix/

>> marketingteacher.com/category/marketing-mix/

> See Guy Kawasaki's blog for **more information on "The Art of Evangelism"**:

>> blog.guykawasaki.com/2006/01/the_art_of_evan.html

6) Profit & Loss (P&L) statement and evaluation

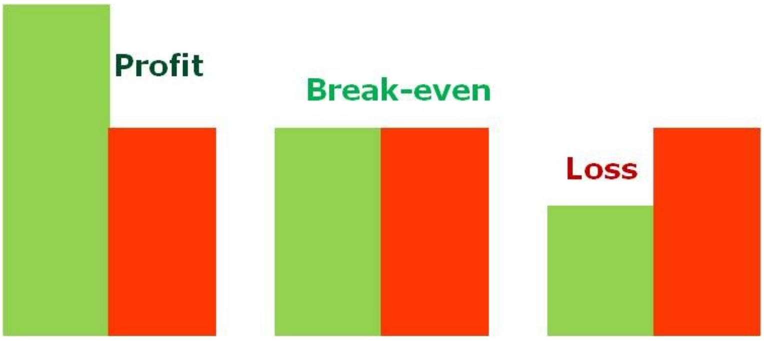
General remarks

This chapter provides an overview of the notion of "**P&L**", which is mentioned on several occasions in this guide.
A P&L evaluation makes it possible to form a judgment on a project's **profitability**.

The basic principle is very simple: the "**bottom line**" of the **P&L statement** is the **net margin**, ie the difference between total expected revenue from a product and the total cost of producing it (in the broadest sense, ie creation + manufacturing), promoting it and selling it; if the net margin is **positive**, then the project is **profitable**; a **negative** net margin corresponds to a **loss**.

The "**break-even**" point is reached when revenues cover costs. This may occur only after some time, at which point the P&L starts to show a profit instead of a loss.

In the following illustration, the left-hand bars represent revenues and the right-hand bars represent costs.



Revenues

Revenues may be generated by **selling products** or **services** or **subscriptions** and/or by **advertising** (eg websites featuring ads).

Revenues may be derived from **licence fees** or **royalties**, or from **commissions** on third-party product sales or transactions. Examples of such a model are eBay and PayPal.

Revenues may take the form of **donations** that are made by users or sponsors or benefactors in order to keep a product alive (eg Wikipedia).

Revenues may also correspond to the **cost reduction** of a business operation over a certain period of time as a result, for example, of the installation and use of a new application or system. The amount of money saved by the cost reduction may however be difficult to evaluate…

> See the following site for **an overview of business models on the web**:

>> digitalenterprise.org/models/models.html

From a contractor's viewpoint, if the project consists in a service provided to a client, the revenues correspond to the **fee paid by the client** for the provision of the service, ie the **price of the service**.
The fee may be divided into a **flat fee** and a **proportional fee** (eg a percentage of gross or net revenues generated by the product). An additional fee is often charged by the contractor for maintenance (unless maintenance is included in the flat fee).

A developer may also charge a **licence fee** for the use of its proprietary technology.

Note that a contractor's fee generally includes a **profit margin**. The price paid by the project owner (the client) is therefore the sum of the contractor’s cost and its profit margin. Another way of putting this is to say that the contractor applies a “**mark-up**” to its cost in order to make a profit.
(There is often **confusion between margin and mark-up expressed as a percent value**: see the section at the end of this chapter for more details on the subject.)

Contractors may be obliged to pay **penalties** to a client, eg in the case of a late delivery. Such penalties may be considered as **revenues** by the client but they are obviously **offset by the cost of damages** caused.

Costs

The **costs** to be taken into account in the P&L are those of the **project** (people and other resources such as equipment, facilities, supplies, service providers, (sub)contractors, travel, etc.) and those of the **resulting product** (manufacturing, maintenance, distribution, marketing, customer services, etc.).

Some costs are **fixed**, for example the project costs; others are **variable**, for example manufacturing and other costs (including possible royalties) which are usually a function of the number of units of the product that are forecasted to be sold.

Project costs will be reviewed in more detail in chapter 9 (“Project planning”).

Product-related costs and revenues are to be estimated by Marketing & Sales, based on the product’s marketing mix as well as information about the market and competition.

The P&L is often established for a period of several years, even if profitability is required within a shorter time frame.

Some of the project costs (software development, servers…) may be treated for accounting purposes as an investment to be amortized over a number of years.

Project funding

Project funding is not to be considered as actual revenue but as the “**fuel**” required to get a project started and successfully completed.

Funds may come from the **project owner’s/sponsor’s organization**, from a **bank** in the form of a loan (to be reimbursed of course!) or from **venture-capital** firms or “**business angels**”, who generally expect a **return on investment** (“ROI”).

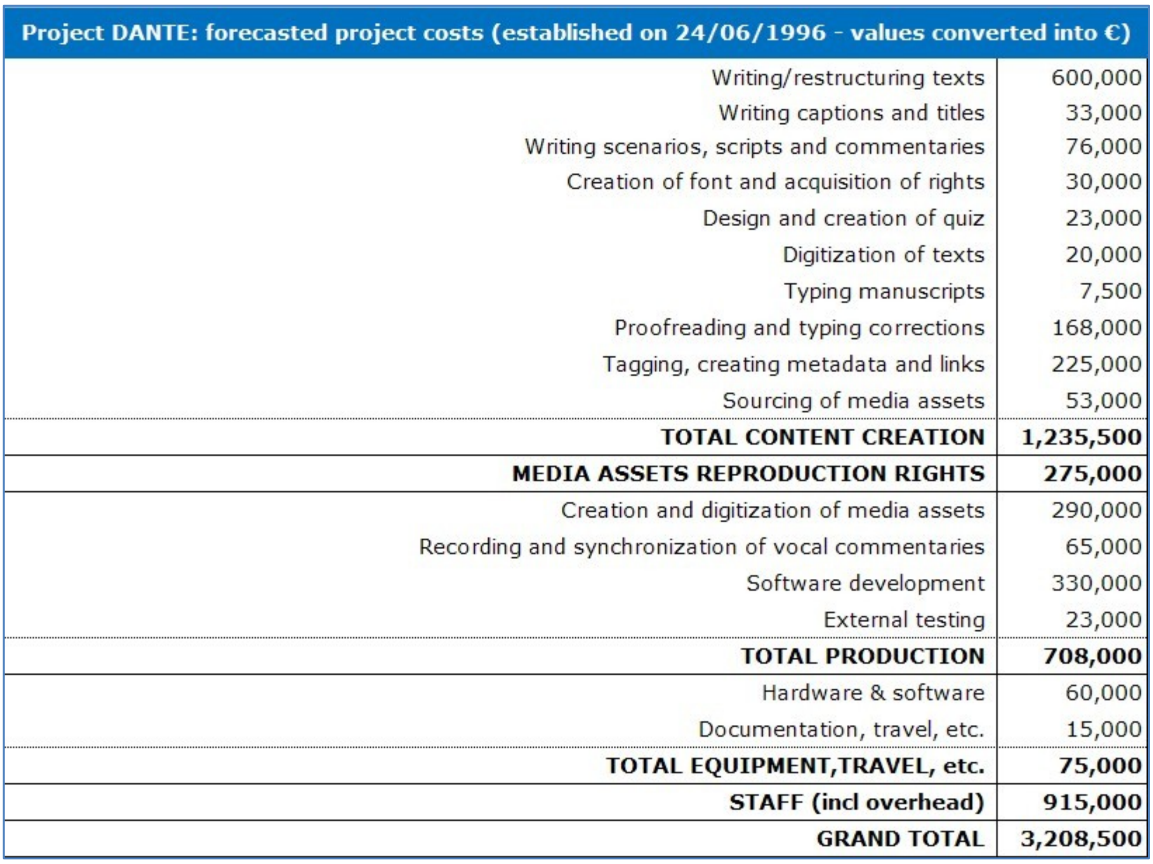
In some cases, the project may be subsidized by an external organization (government, etc.): the **subsidy** may offset all or part of the cost of the project.
Note that some subsidies may be granted on the condition that they should be reimbursed, at least partially, at some stage after the product resulting from the project has been launched and starts to generate revenue.

> *My team at Hachette Multimedia designed and developed two series of “Learning Objects” for schools. 50% of the cost of the first series was funded by the European Commission. 30% of the cost of the second series was funded by the French Ministry of Education. The first project was profitable, while the second project was not, due to a premature interruption of sales of the resulting products (as a consequence of the closing down of Hachette Multimedia).*

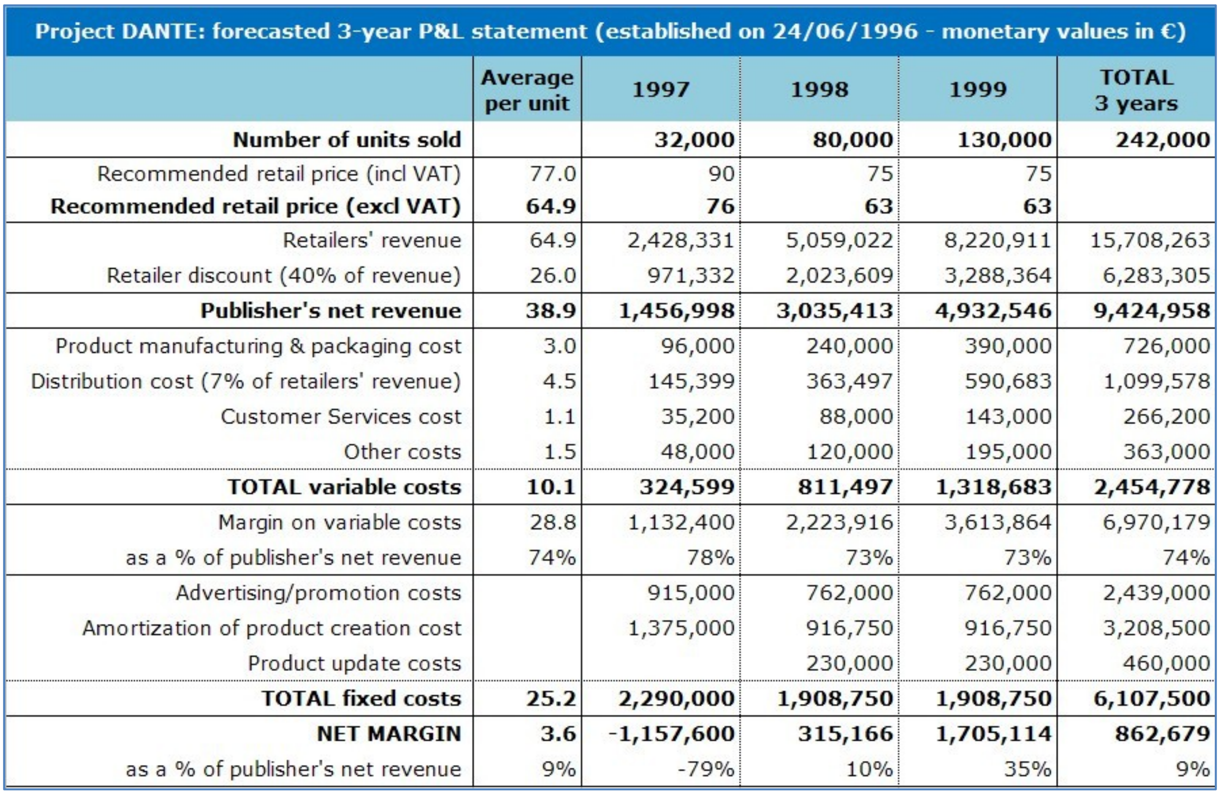
As mentioned above for products, a project may also be funded, at least partially, by **donations** from benefactors of some kind, or by **crowdfunding**.

P&L example

As an example, the following two tables show the forecasted project costs and 3-year P&L statement which were established for the EHM project (code-named "Dante") back in 1996 (they are actually revised versions of the mid-1995 costs and P&L which were featured in the initial "business case").
(Monetary amounts originally in FRF have been converted into EUR.)



Project DANTE: forecasted project costs (established on 24/06/1996 - values converted into €)	
Writing/restructuring texts	600,000
Writing captions and titles	33,000
Writing scenarios, scripts and commentaries	76,000
Creation of font and acquisition of rights	30,000
Design and creation of quiz	23,000
Digitization of texts	20,000
Typing manuscripts	7,500
Proofreading and typing corrections	168,000
Tagging, creating metadata and links	225,000
Sourcing of media assets	53,000
TOTAL CONTENT CREATION	**1,235,500**
MEDIA ASSETS REPRODUCTION RIGHTS	**275,000**
Creation and digitization of media assets	290,000
Recording and synchronization of vocal commentaries	65,000
Software development	330,000
External testing	23,000
TOTAL PRODUCTION	**708,000**
Hardware & software	60,000
Documentation, travel, etc.	15,000
TOTAL EQUIPMENT,TRAVEL, etc.	**75,000**
STAFF (incl overhead)	**915,000**
GRAND TOTAL	**3,208,500**



Project DANTE: forecasted 3-year P&L statement (established on 24/06/1996 - monetary values in €)

	Average per unit	1997	1998	1999	TOTAL 3 years
Number of units sold		**32,000**	**80,000**	**130,000**	**242,000**
Recommended retail price (incl VAT)	77.0	90	75	75	
Recommended retail price (excl VAT)	**64.9**	**76**	**63**	**63**	
Retailers' revenue	64.9	2,428,331	5,059,022	8,220,911	15,708,263
Retailer discount (40% of revenue)	26.0	971,332	2,023,609	3,288,364	6,283,305
Publisher's net revenue	**38.9**	**1,456,998**	**3,035,413**	**4,932,546**	**9,424,958**
Product manufacturing & packaging cost	3.0	96,000	240,000	390,000	726,000
Distribution cost (7% of retailers' revenue)	4.5	145,399	363,497	590,683	1,099,578
Customer Services cost	1.1	35,200	88,000	143,000	266,200
Other costs	1.5	48,000	120,000	195,000	363,000
TOTAL variable costs	**10.1**	**324,599**	**811,497**	**1,318,683**	**2,454,778**
Margin on variable costs	28.8	1,132,400	2,223,916	3,613,864	6,970,179
as a % of publisher's net revenue	74%	78%	73%	73%	74%
Advertising/promotion costs		915,000	762,000	762,000	2,439,000
Amortization of product creation cost		1,375,000	916,750	916,750	3,208,500
Product update costs			230,000	230,000	460,000
TOTAL fixed costs	**25.2**	**2,290,000**	**1,908,750**	**1,908,750**	**6,107,500**
NET MARGIN	**3.6**	**-1,157,600**	**315,166**	**1,705,114**	**862,679**
as a % of publisher's net revenue	9%	-79%	10%	35%	9%

There are circumstances where products are not (or no longer) profitable but nevertheless have a positive impact on a company's business. Such impact is generally "**intangible**" but it is sometimes possible to assign a monetary value to it (though this is not something that financial controllers like!).

> *The EHM, Hachette Multimedia's "flagship of the fleet", was profitable for many years before it started to show a loss. Despite this negative factor, top management at Hachette decided to prolong the product's life for several years, because of its quality and positive contribution to the company's image.*

Extent of a PM's P&L responsibility

The question of whether the **PM** should have "**full P&L responsibility**" (ie should be responsible for the bottom line of the P&L) is a subject of controversy. There are several schools of thought on the matter.
There is no single answer to that question, since the scope of the PM's responsibilities depends on the context and organization of the project. If the PM works for a project execution contractor, eg a software development company, it is likely that he will be given full responsibility for the contractor's P&L, whereas the PM working for a project owner usually shares P&L responsibility with other stakeholders, eg Marketing & Sales. However that may be, the **PM** should at least be **accountable** for a major part of the P&L, namely **project-related costs**.

Margin vs Mark-up

The (profit) **margin** is often expressed as a **percent** value applied to the **price** of a product or service, calculated as follows.

- If P is the price, C the cost, and **Margin%** the percent value of the margin, then…
    - **Margin% = (P - C) / P**
- Price (P) can therefore be determined from cost (C) and Margin% as follows.
    - **P = C / (1 - Margin%)**

For example, if the cost is 75,000 and a margin of 25% is required (Margin% = 25% = 0.25), then the resulting price is 100,000 (= 75,000 / (1 - 0.25) = 75,000 / 0.75).

Another approach for calculating a price consists in applying a **mark-up** to the cost of a product or service. The mark-up is often expressed as a **percent** value applied to the **cost**, calculated as follows.

- If P is the price, C the cost and **Mark-up%** the percent value of the mark-up, then…
    - **Mark-up% = (P - C) / C**
- Price (P) can therefore be determined from cost (C) and Mark-up% as follows.
    - **P = C x (1 + Mark-up%)**

For example, if the cost is 75,000 and a mark-up of 33.3333…% is required (Mark-up% = 33.33…% = 0.3333…), then the resulting price is 100,000 (= 75,000 x 1.3333…).

Whatever the approach, the **profit margin's absolute value** is P – C.

In the above example, P – C = 25,000, ie 1 /4 of the price (Margin% = 0.25 = 25%) and 1/3 of the cost (Mark-up% = 0.3333… = 33.33…%).

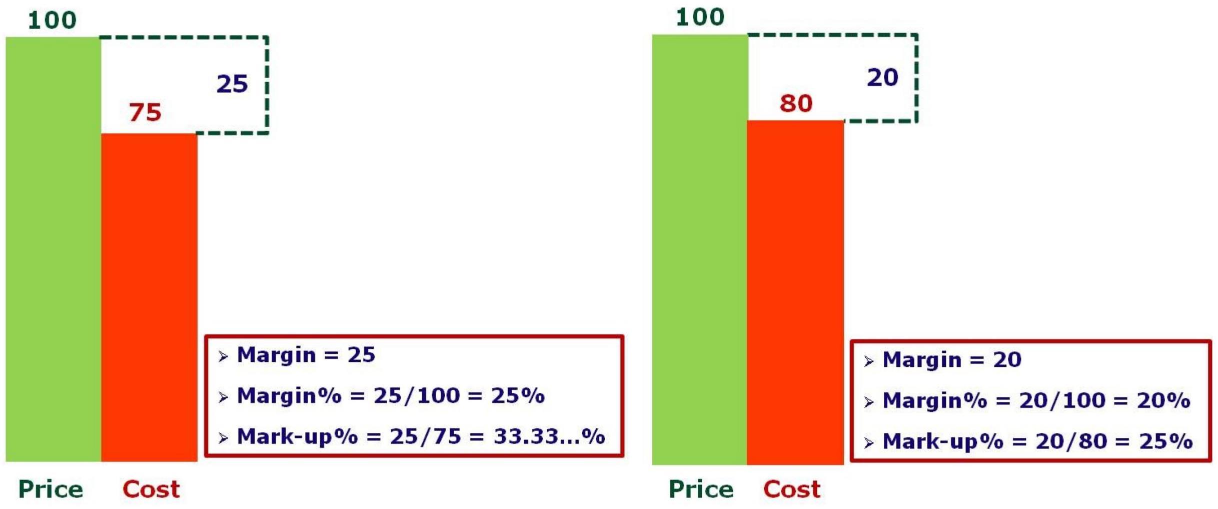
The **relationship between the mark-up and the margin**, both expressed **in percent values**, is given by the following formula.

- **Mark-up% = 1 / (1 – Margin%) – 1**

For example, if the required margin percentage is 25% (Margin% = 25% = 0.25), then the mark-up percentage to apply to cost is 33.33…%.
(1 / (1 - 0.25) – 1 = 1 / 0.75 – 1 = 4/3 – 1 = 1/3 = 0.3333…).

Likewise, if the required margin percentage is 20% (Margin% = 20% = 0.20), then the mark-up percentage to apply to cost is 25%.
(1 / (1 - 0.2) – 1 = 1 / 0.8 – 1 = 5/4 – 1 = 1/4 = 0.25).

The two examples given above are illustrated in the following diagram.



> See the following sites for **more information on the subject**:

>> en.wikipedia.org/wiki/Gross_margin

>> www.accountingtools.com/questions-and-answers/what-is-the-difference-between-margin-and-markup.html

7) Project charter

General remarks

The project charter, also known as project mandate, project definition, project overview statement (POS), project initiation document (PID), terms of reference (TOR) of the project, is a **reference document** that provides a **detailed description of a project**.

The project charter may constitute the **foundation** for a Request for Information (RFI) or Request for Proposal (RFP) issued by the project owner. It may also be used as a basis for the contracts between the project owner and the chosen project implementation entities (including the entity in charge of project implementation).

It generally defines the **scope of the PM's responsibilities** and usually needs to be authorized by whomever the PM is accountable to (even though the business case has already been approved).

Once the project charter is signed, the PM is **officially invested** with the responsibility for the project.

The project charter may be derived from the business case, but it should be more **comprehensive**, in particular as regards the description of the target product and related deliverables.

The project charter should feature at least the items presented below; it should also include (possibly as an appendix) or refer to a **summary requirements specification** for the product to be created (this topic will be addressed in more detail in the next chapter).

Name of the project

First of all, the project should be given a name (usually a code name) in order to make it easy to refer to the project as well as to its team.

The code name may also be used to refer to the product that will result from the project. The definitive name of the product will be chosen in due course by Marketing (possibly with input from the PM).

> *Here are the code names that were used for the three major versions of the Hachette Multimedia Encyclopedia (EHM):*
>
> - *Dante (for the initial version),*
> - *2K (for version 3, dated "2000", a complete redesign and redevelopment of the initial product),*
> - *Eureka (for version 8, dated "2005", another complete redesign and redevelopment).*
>
> *At Apple, the next step was to produce T-shirts bearing the name of the project.* ☺

Objectives and scope of the project

As a foreword or introduction, a brief description of the **project's objectives** should be given: how it fits in with the company's strategy, the needs that will be met and the benefits that will be provided by its outcome, the target product, etc.

A brief presentation of the major **stakeholders** should also be provided (stakeholders being individuals and/or organizations actively involved in the project and/or who have a particular interest in it).

The **scope** of a project, which sets its boundaries, is broadly defined by its **deliverables**. The final (target) deliverable is generally a product, but it could be a service, a new organization, a new process, etc.

As mentioned in chapter 1, it is generally useful to also define what is "**outside scope**", ie what will not be included in the project, in other words what will be explicitly **excluded from the project** (although such a definition can obviously never be exhaustive…).

Assumptions and risks

At this stage of a project, assumptions need to be made; they should be clearly highlighted in the project charter.
Any identified risks, which may be related to or independent of the assumptions, must also be documented. Risks will need to be expanded upon when the detailed project plan is prepared.

Summary project plan

The project charter should include a summary project plan, covering:

- how the various project tasks will be organized: the "work breakdown structure" (WBS),
- the resources required for the project,
- how the project team will be organized,
- the roles and responsibilities of the persons involved,
- the reporting structure,
- the overall timescale and schedule of the project, with its major milestones,
- the project's budget.

These matters will be addressed in more detail in subsequent chapters of this guide.

Since detailed planning of the project is generally undertaken after the project charter has been officially approved, some of the items in the project charter may not be absolutely definitive, for example the detailed organization of the project, intermediate milestones and budget details.
A certain degree of **flexibility**, a **safety margin** (margin of error) and a **provision for risk** should therefore be featured in the project charter, in particular as regards the **schedule** and the **budget**.
One of the **major milestones** usually featured in the project charter, for which there may be some flexibility or none at all, is the date set for the resulting product's launch.

Product launch date

The product launch date, which is merely an objective at this stage of the project, is obviously **fundamental**, since it determines the date at which the project must be completed, which of course has an impact on the resources to be allocated to the project.

The product launch date is also the starting point for "**backward planning**".

The launch date is generally determined as a **trade-off** accepted by the major stakeholders, for example, between what Marketing & Sales want and what the PM considers realistic. In some cases however, the project launch date may be imposed on the PM by the project owner/sponsor (management or the client).

> See the following site for **more information on the Project charter**:

>> https://www.pmhut.com/how-to-write-a-project-charter-part-1

8) Requirements specification

General remarks

The requirements specification is a **key reference document**. It **describes** what the project owner/sponsor expects **the outcome of the project** to be (generally a product, as far as we are concerned here). It should express **user needs and other requirements** in such a way that potential project implementation entities can easily understand the **scope and constraints** of the project and respond to the requirements with a detailed plan, including a cost estimate, and, at a later stage, with detailed design specifications of the target product.

As mentioned in the previous chapter, a summary requirements specification is usually provided as part of, or appended to, the project charter, but a more elaborate and complete requirements specification needs to be written before detailed project planning can be undertaken.

If project execution is outsourced, the requirements specification is usually appended to the contract between the project owner and the entity in charge of project implementation (sometimes called "main contractor" or "prime contractor").

Subsets of this reference document may be used, possibly with some rewriting, for particular requirements specifications intended for (sub)contractors to which parts of the project work may be given.

The requirements specification document intended for contractors is sometimes referred to as the "**Scope statement**" or "**Statement of Work (SOW)**".

There are several methods for **collecting requirements** from future users and other project stakeholders: one-to-one or group interviews, focus groups, workshops, brainstorming sessions, questionnaires, etc.

The PM (or whoever is assigned to writing the requirements document) **manages the process of collecting "input" and consolidating it**.
Marketing & Sales ("M&S") should be involved in the requirements specification process, and sometimes provides the PM with a preliminary expression of needs.
(See chapter 18 for more information on cooperation between the PM and M&S.)

In theory, according to some product development life cycle models, the requirements specification should not be modified during the project's execution phase. This recommendation is however difficult to follow in "real life". More pragmatic methodologies, such as Agile Software Development, provide more **flexibility** and even encourage the revision of requirements as implementation moves forward.
A number of development life cycle models are presented in chapter 12 of this guide.

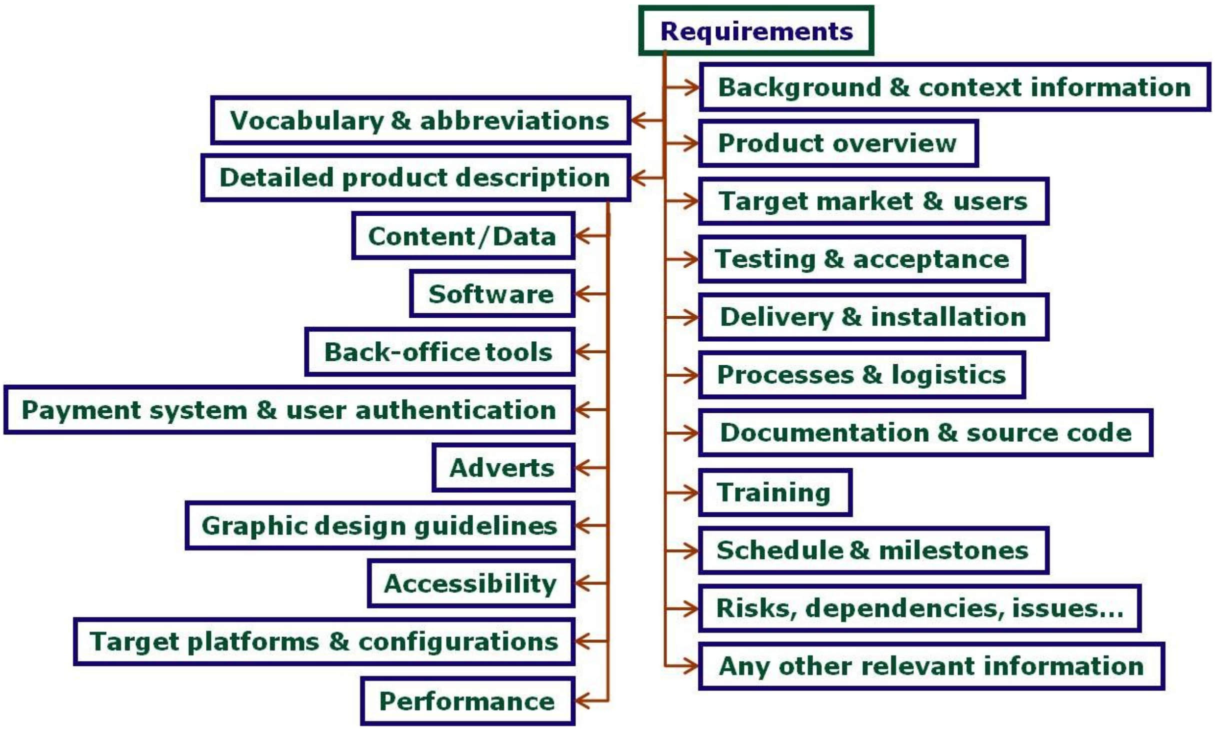
As an example of "built-in flexibility", which is generally desirable, here is an excerpt from the introduction chapter of a requirements specification written by a British publishing company (in 2007):

> *"We have provided as full and comprehensive a description of our requirements as we are currently able, but we do not consider this specification to be complete and final. We expect to work collaboratively with [the Developer] during the design phase for the products to refine and finalize these requirements, where necessary."*

The initial requirements specification and its various revisions must be approved by all project stakeholders concerned.

The **final version of the requirements specification** is a **baseline document** that can be used as a **benchmark for the acceptance testing** of the finished product.

A summary (non-exhaustive) description of the **main components of a requirements specification**, from which subsets may be derived as needed, is given hereafter, assuming that the target product includes content (data), to be provided by the project owner (or an editorial subproject implementation entity), and software (which may be a website), as well as back-office (editing and administration) tools, to be developed by the technical subproject implementation entity.
The list of components may obviously vary according to the specific nature of a project.


Requirements
Background & context information
Vocabulary & abbreviations
Product overview
Detailed product description
Target market & users
Content/Data
Testing & acceptance
Software
Delivery & installation
Back-office tools
Processes & logistics
Payment system & user authentication
Documentation & source code
Adverts
Training
Graphic design guidelines
Schedule & milestones
Accessibility
Risks, dependencies, issues...
Target platforms & configurations
Any other relevant information
Performance

Background information

If the requirements specification is to be used as a stand-alone document, for example for discussion and contractual agreement between the project owner and an external project implementation entity, it should include appropriate excerpts from the project charter document, in order to clearly explain the **overall context of the project**.

Such excerpts may relate to some of the following topics:

- context and objectives of the project,
- presentation of the project owner and other stakeholders,
- business model (insofar as it has an influence on the product requirements).

Vocabulary/abbreviations

It is useful to provide, at the beginning of the requirements specification or as an appendix, a list of specific vocabulary (with precise definitions) and abbreviations used in the document, in order to avoid any ambiguity and misinterpretation.

Product overview

A **summary description of the product**, preferably with a **graphical representation** if possible, is useful as an introduction to the more detailed information that will be provided in subsequent sections of the specification.
This overview should address the "problem" that the product is supposed to solve, the proposed solution and the benefits it is expected to provide to its users.

Target market and users

This section should describe the target market and users of the product. The typology of users may indeed have an influence on the product's functionality and design.
Some of the functions and features of a product may be available to all users, while others may be reserved for certain categories of user, for example administrators, webmasters or editors.

The design of a product may need to take into account the age range of users (children, adults, senior citizens…) or the market segment to which they belong (general public, business, education, government…).

The requirements document may include, in this section or in the "Software" section, so-called "**user stories**" or "**use cases**", which are short descriptions of product functions and features from the viewpoint of a user (or category of user).

Detailed product description

Content / Data

A product's content (or data) may consist of **text and multimedia assets**, the latter being still images, animations, videos and sound. If the product is a business application, its **data** may be restricted to **text and numbers** (amounts, dates, etc.), and possibly **pictures, diagrams and charts**.

Content needs to be described at the level of detail required for product development.
A section or an appendix may actually be dedicated to the specification of the "**data delivery format**", ie the format in which data will be delivered by the content provider to the software developer.

If content is expected to be **multilingual**, the languages to be supported and possible character set constraints (fonts, encoding) must be clearly specified. More generally, clear and comprehensive **guidelines for localization** should be provided.

It may be useful to group content elements by **type**, each type possibly treated as a specific editorial subproject.

The EHM's content was broken down into the following "data sets":

- *encyclopedic articles (the fundamental content of the encyclopedia),*
- *definitions (French-language dictionary),*
- *summary articles (for schoolchildren),*
- *document excerpts (literature, speeches, etc.),*
- *atlas (maps and other geography-related media, including data sheets for all countries),*
- *photographs and drawings,*
- *animations (some interactive),*
- *videos,*
- *sound documents,*
- *timeline (a graphical panorama and related notes),*
- *facts & figures tables,*
- *website URLs and descriptions (linked to encyclopedic articles),*
- *media captions,*
- *quiz.*

For each data type, the (approximate) **number** of items, their estimated **size** ("footprint") and **format** should be documented.

> *The content of the last edition of the EHM on DVD-ROM included approximately 108,000 texts (42,000 articles and 66,000 definitions), "weighing" 100 million or so characters, the texts being illustrated by close to 17,000 multimedia assets on the DVD-ROM (roughly half that amount in the online EHM), some of which, in particular 3D rendering of historic sites, with a footprint of several megabytes each.*
>
> *Formats were: JPEG for still images, QuickTime for videos and non-interactive animations, Flash for interactive animations and interactive maps, QuickTime (DVD-ROM version) and MP3 (online version) for sound.*

Content that is visible to the end user is generally only part of the full set of data in a product. **Tags** and, more generally, **metadata** are usually required, for example for searching (indexes), for content display (style sheets, etc.), as well as for links between elements of content.
Note that the input of metadata and the setting of links, which are often manual operations, are editorial tasks that can be very heavy and time-consuming.

> *For the EHM, structured text was marked up in XML (initially in SGML), using a very complex "DTD" (Document Type Definition), which was provided as an appendix to the requirements specification.*
> *Furthermore, there were metadata associated to each "document" (the generic name use to refer to an item of content: article, definition, media asset…), and, in the case of long articles or definitions of words with multiple senses, to document sections or subsections. The metadata included one or several topics (subjects), chosen from a "topic tree" with close to 3,200 "leaves", and, if applicable, one or several geographical locations (regions, countries…) and periods (dates), such metadata being used for a number of functions, in particular advanced search.*
> *Finally, each multimedia asset was manually linked to one or several texts.*

There is also content that may be called "**secondary**" or "**ancillary**" but which is **nevertheless important** and has to be taken into consideration for both editorial and design/development/integration purposes.
Such content may be for example: help text, credits, licencing and other legal information, sales terms & conditions, contact information, etc.

> *For example, the online EHM's ancillary content included:*
>
> - *home page text and illustrations,*
> - *conjugation tables,*
> - *a pedagogical guide,*
> - *"Help", integrated in the software application,*
> - *"About…" (credits, etc.),*
> - *"User licence" and other legal information,*
> - *commercial pages (presentation of the product offering and fees, subscription process…).*

Note that **elements of the user interface** of a product (eg menus, menu items, dialog boxes, text buttons, tooltips, error/warning messages, etc.) and **metadata** should also be considered as **content** since they need to be created by people with editorial skills.

In the case of a "static" application (eg a non-dynamic website), information regarding content may be complemented with an indication of the number and size of "pages".

If applicable, the specification should also address **content updates and additions**: which areas/items of content are liable to be modified or expanded, to what extent and how frequently.

For a website, more or less simple "**holding pages**" may be required, their content being intended to "whet the appetite" of visitors before the final product actually goes online. Such pages also need to be mentioned in the requirements specification.

Software

The software (in the broadest sense of the word, which includes websites and web-based applications or "cloud" applications) exploits content and makes it available to users by means of a **user interface** and a set of **functions and related features**.

The **description** of the software in the requirements specification may be given at a **high level** (ie not at a very detailed level) but it should nevertheless be **exhaustive**, in that it should encompass **all of its functions and features**, as well as the **main characteristics of its user interface**. The objective is to enable readers to clearly understand what the software should achieve. In this respect, a **graphical representation** of the software's functionality (and mock-up user interface), as well as evocative **metaphors**, may help to make a long narrative text easier to understand.

It is sometimes possible to rank functions and features by **priority**, for example in two groups: "**must-have**" and "**nice-to-have**" (ie "desirable but not absolutely necessary"). Doing this allows the implementation entity to divide the response to the requirements into two distinct sets of functions and features, each with a specific price attached.

Any foreseeable **evolution** of the software's functions and features, as well as any requirement for **subsets** of the product or **by-products**, should also be documented.

> *For the most recent version of the online EHM, the software's main functions and features could be described as follows:*
>
> - *online multimedia encyclopedia dynamic website, exploiting a content database, with frequent updates and additions,*
> - *search box always present in the upper part of the interface,*
> - *standard search (full text, multiple words, with "wildcard" representing any number of characters),*
> - *presentation of standard search results in "panorama" mode, comparable to the front page of a newspaper, in three areas, the central area occupying half of the total width of the page and dedicated to the preview of most "relevant" results and related media, the left and right columns being reserved for previews and links to related documents,*
> - *presentation of standard search results in "list" mode upon user request,*
> - *presentation of a document, selected from the panorama or from the list, in the central column, the left and right columns being reserved for previews and links to related documents,*
> - *long articles on a single page with table of contents in the left column,*
> - *full-width display of documents as a user-triggered option,*
> - *advanced (multicriteria) search (by topic, location, period, words, with AND, OR, NOT boolean operators),*
> - *presentation of advanced search results in "list" mode (the panorama mode being irrelevant for this type of search),*
> - *hypertext via double click on any word (launches a standard search),*
> - *access by single click to "Help" and "About…",*
> - *access by single click to the conjugation tables and pedagogical guide,*
> - *compliance with the W3C/WAI recommendations concerning accessibility for visually-impaired users,*

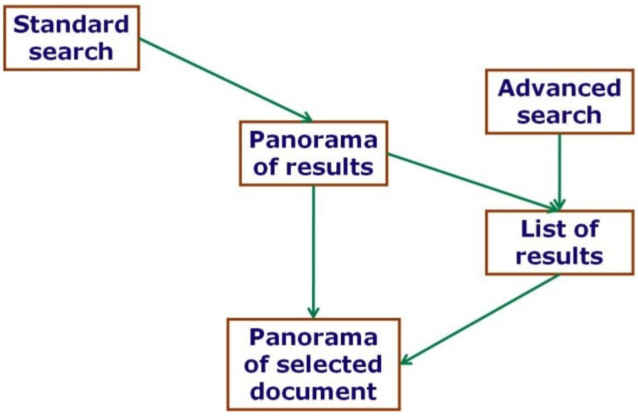
- *subset of the encyclopedia (home page, "help", "about…", standard search and panorama of results) available free of charge,*

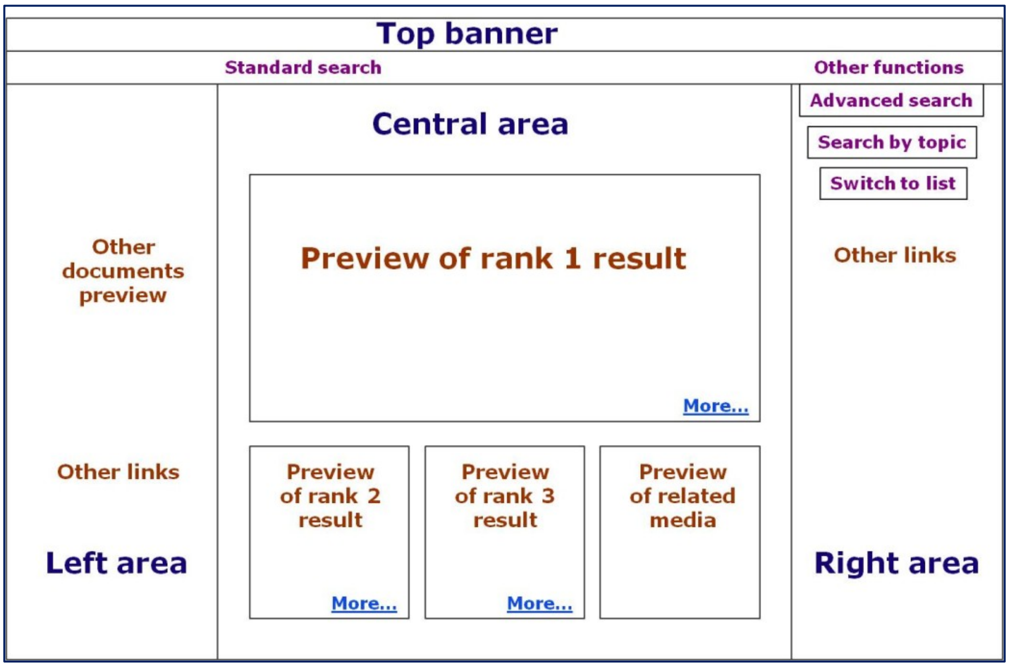
- *full content and functionality available by subscription,*

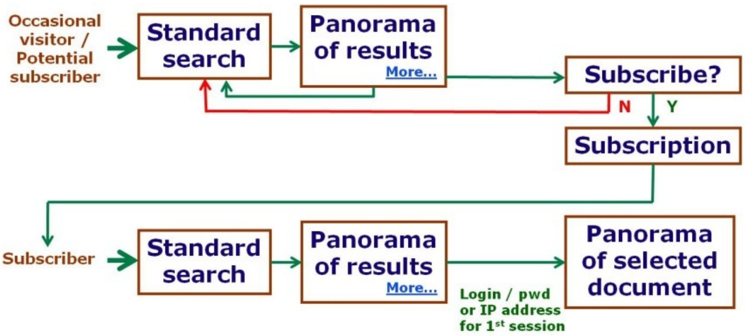
- *payment by "Audiotel" (toll phone) or by credit card,*

- *authentication by login/password or by IP address.*

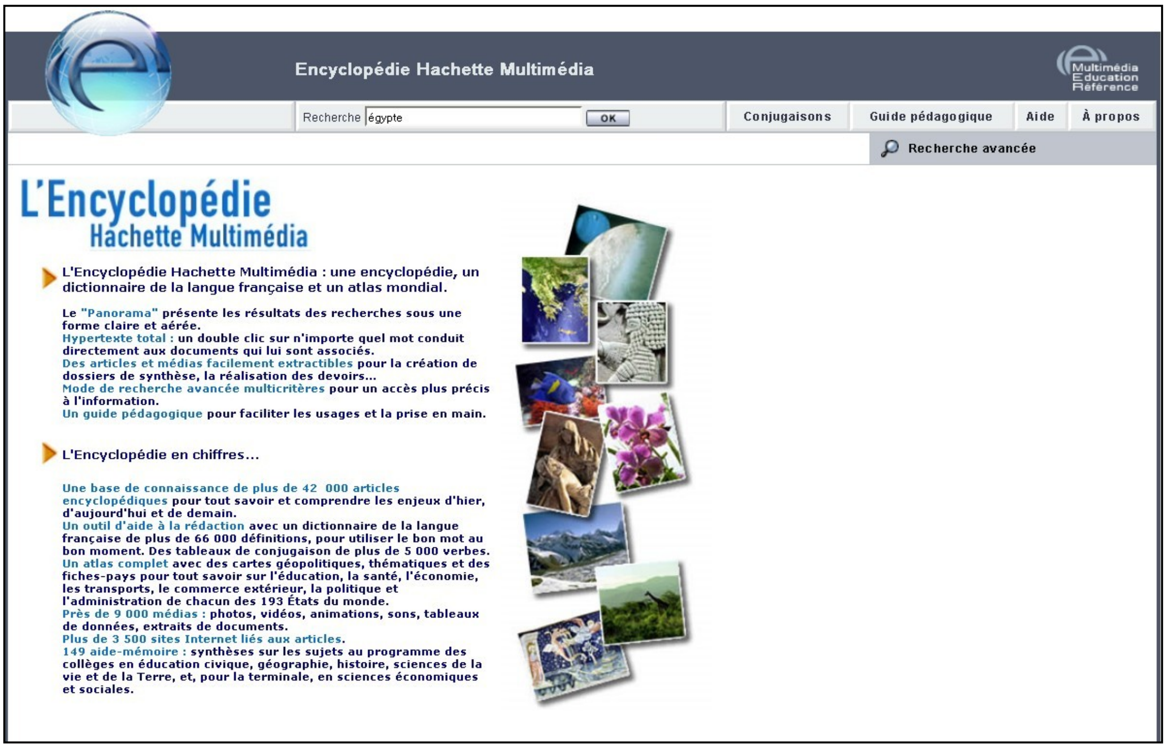
Below are examples of block diagrams and graphical representations of some of the above functions and features of the EHM, including an illustration of the "panorama" metaphor (reminding of the front page of a newspaper), as featured in the requirements specification, followed by screenshots showing how the functions and features were implemented.







Search the EHM (free of charge):



L'Encyclopédie Hachette Multimédia

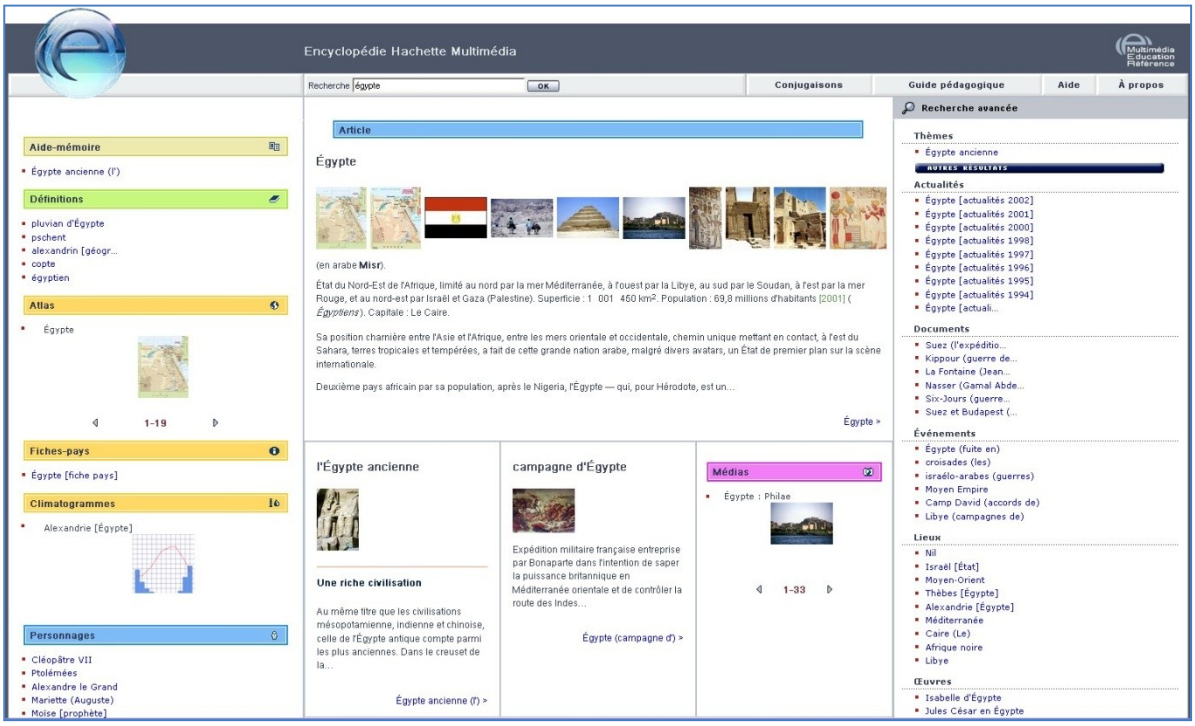
L'Encyclopédie Hachette Multimédia : une encyclopédie, un dictionnaire de la langue française et un atlas mondial.

Le "Panorama" présente les résultats des recherches sous une forme claire et aérée.
Hypertexte total : un double clic sur n'importe quel mot conduit directement aux documents qui lui sont associés.
Des articles et médias facilement extractibles pour la création de dossiers de synthèse, la réalisation des devoirs...
Mode de recherche avancée multicritères pour un accès plus précis à l'information.
Un guide pédagogique pour faciliter les usages et la prise en main.

L'Encyclopédie en chiffres...

Une base de connaissance de plus de 42 000 articles encyclopédiques pour tout savoir et comprendre les enjeux d'hier, d'aujourd'hui et de demain.
Un outil d'aide à la rédaction avec un dictionnaire de la langue française de plus de 66 000 définitions, pour utiliser le bon mot au bon moment. Des tableaux de conjugaison de plus de 5 000 verbes.
Un atlas complet avec des cartes géopolitiques, thématiques et des fiches-pays pour tout savoir sur l'éducation, la santé, l'économie, les transports, le commerce extérieur, la politique et l'administration de chacun des 193 États du monde.
Près de 9 000 médias : photos, vidéos, animations, sons, tableaux de données, extraits de documents.
Plus de 3 500 sites Internet liés aux articles.
149 aide-mémoire : synthèses sur les sujets au programme des collèges en éducation civique, géographie, histoire, sciences de la vie et de la Terre, et, pour la terminale, en sciences économiques et sociales.

Panorama of results (provided free of charge):



Aide-mémoire
- Égypte ancienne (l')

Définitions
- pluvian d'Égypte
- pschent
- alexandrin [géogr...
- copte
- égyptien

Atlas
- Égypte

Fiches-pays
- Égypte [fiche pays]

Climatogrammes
- Alexandrie [Égypte]

Personnages
- Cléopâtre VII
- Ptolémées
- Alexandre le Grand
- Mariette (Auguste)
- Moïse [prophète]

Article

Égypte

(en arabe **Misr**).

État du Nord-Est de l'Afrique, limité au nord par la mer Méditerranée, à l'ouest par la Libye, au sud par le Soudan, à l'est par la mer Rouge, et au nord-est par Israël et Gaza (Palestine). Superficie : 1 001 450 km^2. Population : 69,8 millions d'habitants [2001] (*Égyptiens*). Capitale : Le Caire.

Sa position charnière entre l'Asie et l'Afrique, entre les mers orientale et occidentale, chemin unique mettant en contact, à l'est du Sahara, terres tropicales et tempérées, a fait de cette grande nation arabe, malgré divers avatars, un État de premier plan sur la scène internationale.

Deuxième pays africain par sa population, après le Nigeria, l'Égypte — qui, pour Hérodote, est un...

Égypte »

l'Égypte ancienne

Une riche civilisation

Au même titre que les civilisations mésopotamienne, indienne et chinoise, celle de l'Égypte antique compte parmi les plus anciennes. Dans le creuset de la...

Égypte ancienne (l') >

campagne d'Égypte

Expédition militaire française entreprise par Bonaparte dans l'intention de saper la puissance britannique en Méditerranée orientale et de contrôler la route des Indes...

Égypte (campagne d') >

Médias
- Égypte : Philae

Thèmes
- Égypte ancienne

Actualités
- Égypte [actualités 2002]
- Égypte [actualités 2001]
- Égypte [actualités 2000]
- Égypte [actualités 1998]
- Égypte [actualités 1997]
- Égypte [actualités 1996]
- Égypte [actualités 1995]
- Égypte [actualités 1994]
- Égypte [actuali...

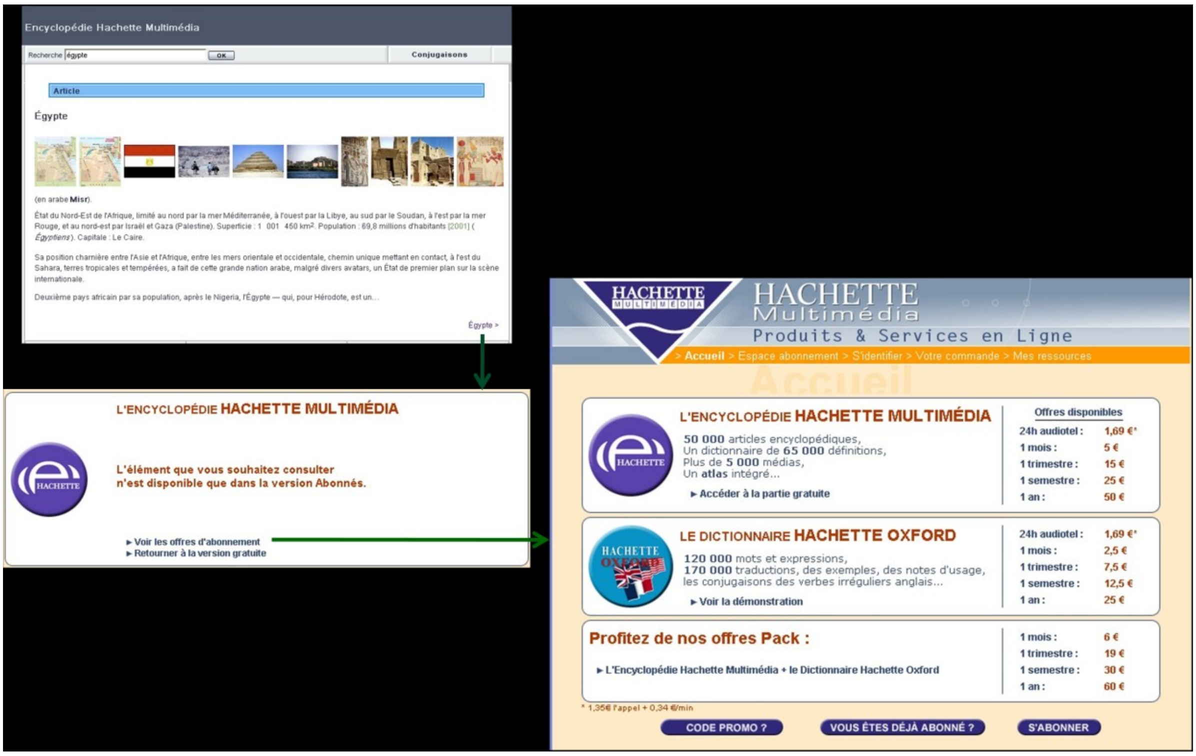
Documents
- Suez (l'expéditio...
- Kippour (guerre de...
- La Fontaine (Jean...
- Nasser (Gamal Abde...
- Six-Jours (guerre...
- Suez et Budapest (...

Événements
- Égypte (fuite en)
- croisades (les)
- israélo-arabes (guerres)
- Moyen Empire
- Camp David (accords de)
- Libye (campagnes de)

Lieux
- Nil
- Israël [État]
- Moyen-Orient
- Thèbes [Égypte]
- Alexandrie [Égypte]
- Méditerranée
- Caire (Le)
- Afrique noire
- Libye

Œuvres
- Isabelle d'Égypte
- Jules César en Égypte

Click on link to display the full content of the main article: if you are not a subscriber, you are given information about the various offers and the option to subscribe or return to the part of the EHM which is free of charge:



Encyclopédie Hachette Multimédia

Recherche égypte OK | Conjugaisons

Article

Égypte

(en arabe **Misr**).

État du Nord-Est de l'Afrique, limité au nord par la mer Méditerranée, à l'ouest par la Libye, au sud par le Soudan, à l'est par la mer Rouge, et au nord-est par Israël et Gaza (Palestine). Superficie : 1 001 450 km². Population : 69,8 millions d'habitants [2001] (*Égyptiens*). Capitale : Le Caire.

Sa position charnière entre l'Asie et l'Afrique, entre les mers orientale et occidentale, chemin unique mettant en contact, à l'est du Sahara, terres tropicales et tempérées, a fait de cette grande nation arabe, malgré divers avatars, un État de premier plan sur la scène internationale.

Deuxième pays africain par sa population, après le Nigeria, l'Égypte — qui, pour Hérodote, est un…

Égypte >

L'ENCYCLOPÉDIE HACHETTE MULTIMÉDIA

L'élément que vous souhaitez consulter n'est disponible que dans la version Abonnés.

► Voir les offres d'abonnement
► Retourner à la version gratuite

HACHETTE Multimédia
Produits & Services en Ligne

> Accueil > Espace abonnement > S'identifier > Votre commande > Mes ressources

Accueil

L'ENCYCLOPÉDIE HACHETTE MULTIMÉDIA

50 000 articles encyclopédiques,
Un dictionnaire de **65 000** définitions,
Plus de **5 000** médias,
Un **atlas** intégré…

► Accéder à la partie gratuite

Offres disponibles	
24h audiotel :	1,69 €*
1 mois :	5 €
1 trimestre :	15 €
1 semestre :	25 €
1 an :	50 €

LE DICTIONNAIRE HACHETTE OXFORD

120 000 mots et expressions,
170 000 traductions, des exemples, des notes d'usage,
les conjugaisons des verbes irréguliers anglais…

► Voir la démonstration

24h audiotel :	1,69 €*
1 mois :	2,5 €
1 trimestre :	7,5 €
1 semestre :	12,5 €
1 an :	25 €

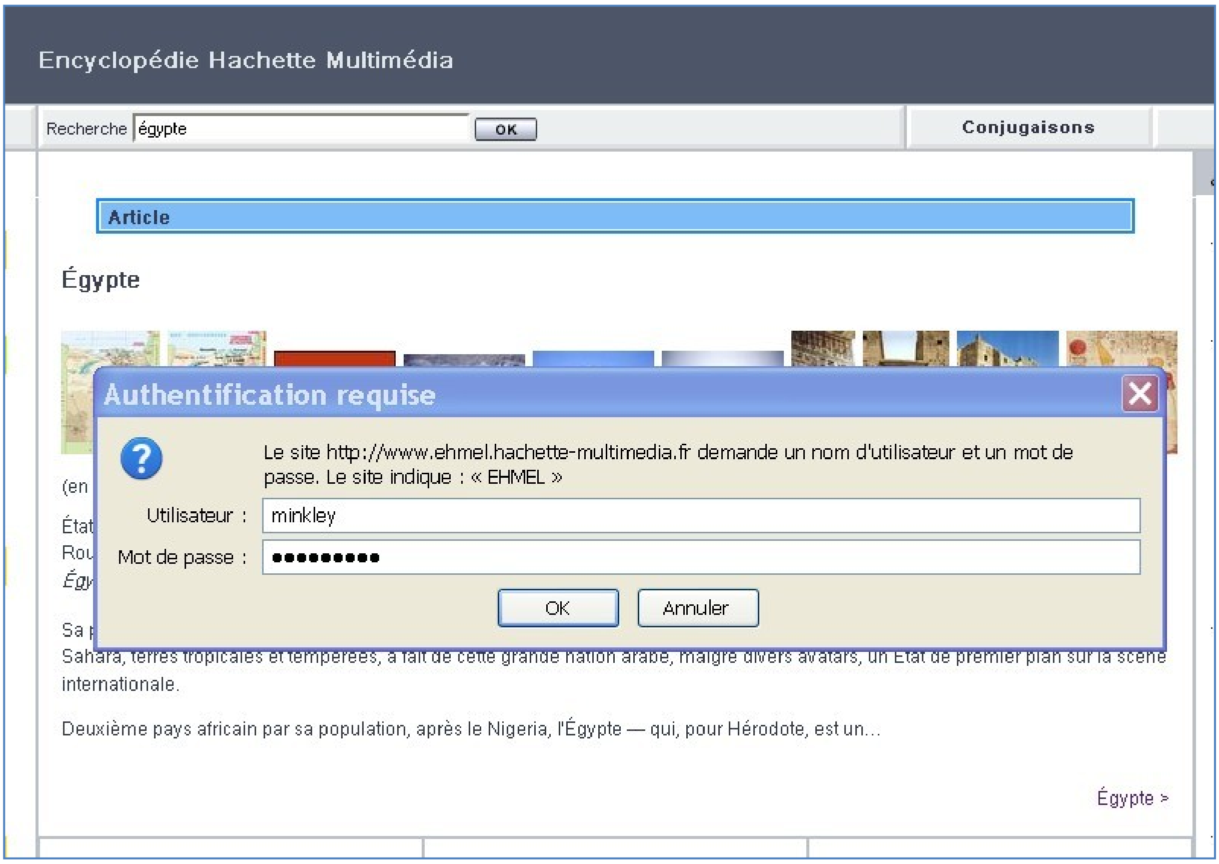
Profitez de nos offres Pack :

► L'Encyclopédie Hachette Multimédia + le Dictionnaire Hachette Oxford

1 mois :	6 €
1 trimestre :	19 €
1 semestre :	30 €
1 an :	60 €

* 1,35€ l'appel + 0,34 €/min

If you have already subscribed to the EHM, you need to go through an authentication process (only once per session):



Encyclopédie Hachette Multimédia

Recherche égypte OK | Conjugaisons

Article

Égypte

Authentification requise

Le site http://www.ehmel.hachette-multimedia.fr demande un nom d'utilisateur et un mot de passe. Le site indique : « EHMEL »

Utilisateur : minkley

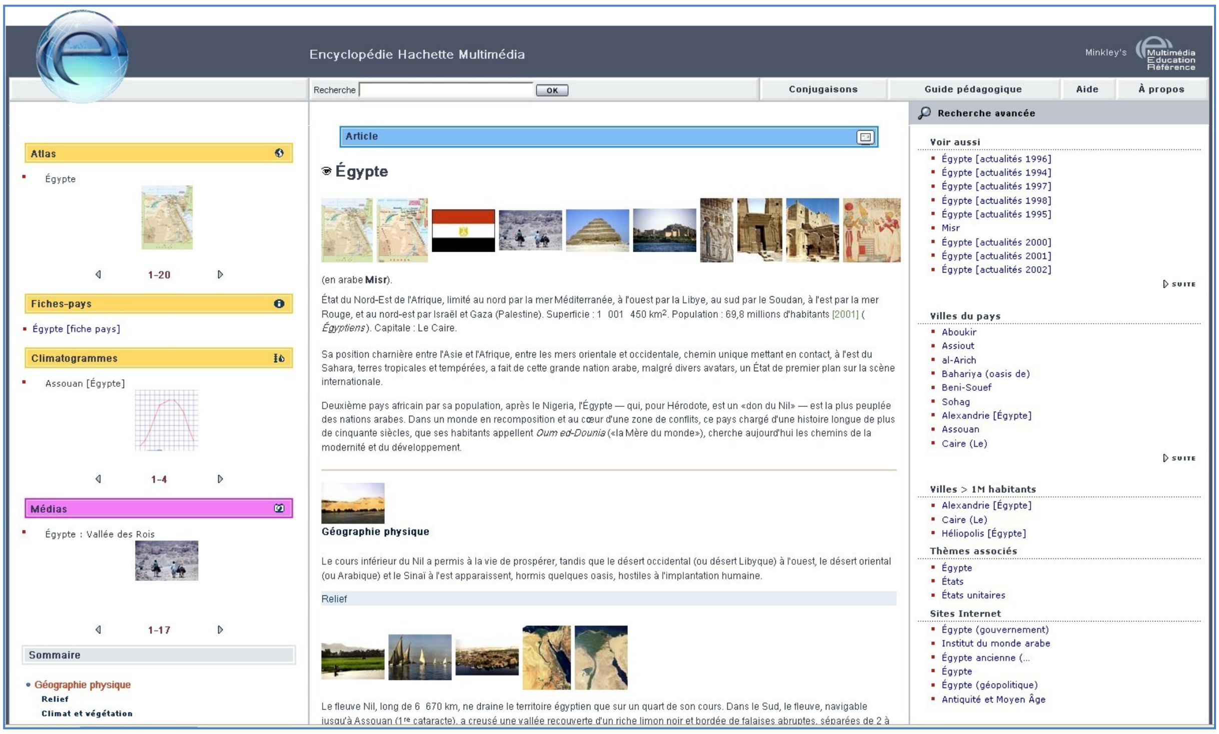
Mot de passe : ●●●●●●●●●●

Sahara, terres tropicales et tempérées, a fait de cette grande nation arabe, malgré divers avatars, un État de premier plan sur la scène internationale.

Deuxième pays africain par sa population, après le Nigeria, l'Égypte — qui, pour Hérodote, est un…

Égypte >

You now have access to the full content of the encyclopedia (starting with the article of which only a preview was displayed as a result of the initial search):



Égypte

(en arabe **Misr**).

État du Nord-Est de l'Afrique, limité au nord par la mer Méditerranée, à l'ouest par la Libye, au sud par le Soudan, à l'est par la mer Rouge, et au nord-est par Israël et Gaza (Palestine). Superficie : 1 001 450 km^2. Population : 69,8 millions d'habitants [2001] (*Égyptiens*). Capitale : Le Caire.

Sa position charnière entre l'Asie et l'Afrique, entre les mers orientale et occidentale, chemin unique mettant en contact, à l'est du Sahara, terres tropicales et tempérées, a fait de cette grande nation arabe, malgré divers avatars, un État de premier plan sur la scène internationale.

Deuxième pays africain par sa population, après le Nigeria, l'Égypte — qui, pour Hérodote, est un «don du Nil» — est la plus peuplée des nations arabes. Dans un monde en recomposition et au cœur d'une zone de conflits, ce pays chargé d'une histoire longue de plus de cinquante siècles, que ses habitants appellent *Oum ed-Dounia* («la Mère du monde»), cherche aujourd'hui les chemins de la modernité et du développement.

Géographie physique

Le cours inférieur du Nil a permis à la vie de prospérer, tandis que le désert occidental (ou désert Libyque) à l'ouest, le désert oriental (ou Arabique) et le Sinaï à l'est apparaissent, hormis quelques oasis, hostiles à l'implantation humaine.

Relief

Le fleuve Nil, long de 6 670 km, ne draine le territoire égyptien que sur un quart de son cours. Dans le Sud, le fleuve, navigable jusqu'à Assouan (1re cataracte), a creusé une vallée recouverte d'un riche limon noir et bordée de falaises abruptes, séparées de 2 à

Back-office (editing and administration) tools

For products that include content which is liable to evolve (through updates and additions) on a regular basis, it is necessary to provide **editors**, who are generally **not IT specialists**, with tools (eg a “content management system”) enabling them to **manage content** (data and metadata input, modification, publication, etc.) in a simple and efficient manner, **without** any need for help from **developers**.

Furthermore, products such as websites generally require some form of **administration**, usually performed by an administrator or webmaster.

The editing and administration tools form what may be called a “**back office**” (or "back end"). These tool sets, the creation of which may actually constitute a **subproject**, must be specified in the requirements.

The editors of the latest versions of the EHM used a back-office system that was custom-made by the company that developed the EHM software. The system included an Oracle database and content management/editing tools that were accessible via an intranet, using a Mozilla-based custom browser.

Another tool was used after each “data freeze” to extract the content of a new edition of the EHM from the Oracle database, to create all of the indexes required for the various types of search, to format the data for the production of a CD/DVD-ROM or a new online version, and, for the latter, to actually publish the content on the EHM website.

The database and editing tools resided on two servers (one mirroring the other as a back-up). The extraction, indexing, formatting and publishing tools resided on a third, high-performance server.

> *All necessary functions of this content management system had been specified in a comprehensive and detailed requirements document which was the consolidated result of interviews of editors conducted by the developers, and of workshops, brainstorming sessions and other meetings involving all persons concerned.*
>
> *The requirements document was actually written by a PM working for the development company, then submitted for review and approval to Hachette (the editors and myself, as PM for the project owner). Needless to say, there were many revisions of the requirements specification before the final version was agreed upon by both parties.*
>
> *The functions provided by the online EHM administration tool set included user account management (creation, modification, deletion) and usage statistics.*

Payment system and user authentication

A payment system may be required as part of the application to be developed. The project owner may have already chosen such a system (PayPal, SWREG, etc.) or may leave the issue open for discussion. However that may be, the payment process and required user authentication mechanism should be documented in the requirements specification, along with the specification of information items that need to be entered by users and stored in some form of database (eg customer information such as name, address, e-mail, login, password, etc.).

A "**single sign-on**" (SSO) mechanism may need to be implemented so that users do not have to identify themselves several times, for example in a situation where a web application requiring authentication is accessed via a website which also requires authentication.

> *The online EHM was made available to schools via a platform (the "Kiosque numérique de l'éducation": www.kiosque-edu.com) to which users (teachers and students) could log in for access to all of the resources to which their school had subscribed, including the EHM, without any further authentication required.*

Some applications do not involve any payment but nevertheless require user authentication (eg an intranet or extranet application, restricted to certain categories of users).
Conversely, there are applications that involve payment but which do not require user authentication for initial access to the website (eg an online store).

Adverts

If adverts are to appear in the product (eg on a website), their desired dimensions and position in the interface should be specified, as well as any requirement or suggestion regarding the advert feed mechanism (eg the use of an "ad server").

Graphic design guidelines

If user-interface graphic design guidelines or recommendations are available, they should be included in the requirements specification.
The distribution of roles in this area should also be clearly indicated. Responsibility for graphic design may be given to the developer or may be held by the project owner, who may choose to subcontract the work to a specialized agency.
Any requirement or preference for fonts and font sizes should be documented.

Mention should be made of logos or other identity-related items (usually supplied by the project owner) which are to be displayed in the user interface in specific locations.

Accessibility

Requirements for compliance of the product with accessibility standards (eg W3C/WAI) need to be clearly specified.

Target platforms and configurations

The hardware and software environments in which the product is expected to function, as well as minimum configurations required for the product to work properly, must be clearly documented.

Such requirements may result from a market study and competitive analysis. They, as well as performance requirements, may of course be challenged by the developers, but once the requirements document has finally been agreed upon by both parties, these requirements become part of the project execution contract that developers must comply with.

> *The latest version of the EHM on DVD-ROM was required to run on configurations specified as follows:*
>
> - *PCs with Windows 98SE, Me, 2000, NT, XP or Vista; PCs with Linux;*
> - *Macintoshes with Mac OS X;*
> - *600-MHz Pentium 3 for PCs (should also work on Pentium 4 and Intel Core Duo PCs);*
> - *500-MHz G3 for Macintoshes (should also work on G4, G5 and Intel Core Duo Macs);*
> - *512 Megabytes of available RAM;*
> - *DVD-ROM drive;*
> - *1 or 2 Gigabytes of available hard disk space (for partial or full installation);*
> - *1024x768 screen resolution.*
>
> *The compatibility requirements for the online EHM were the following:*
>
> - *Windows: Microsoft Internet Explorer 6 (or later), Mozilla Firefox;*
> - *Linux: Mozilla Firefox;*
> - *Mac OS 9: Netscape 7;*
> - *Mac OS X: Safari, Mozilla Firefox;*
> - *Display optimized for a 1024x768 screen resolution.*

The hardware and software infrastructure required for the operation of an information system or product may be unknown to the project owner, for lack of technical knowledge on his part. In such a situation, the project owner usually relies on the project implementation entity to make appropriate recommendations (network architecture and components, server hardware and software configurations, etc.), as part of the response to the requirements specification.

> *The subcontractor used by Hachette for the development of the online version of the EHM determined the type and configuration of the servers required for the product to run properly and with an acceptable level of performance. The recommended servers were procured by Hachette and installed in its hosting service provider's server centre.*

Performance

It is advisable for certain categories of product to specify the minimum speed, throughput or, more generally, the performance of the product to be developed, at least as far as major functions are concerned. Whenever possible, **detailed benchmarks** should be provided in order to measure the degree of achievement of the performance objectives.

The following indications (among others) were given for the first version of the EHM (which was released in March 1998):

- *the scrolling of a long list (including the full list of entries) should be fast and fluid (these "subjective" criteria were not ideal…);*
- *the display of an article should be completed in less than one second after a click on its title in a list;*
- *the display of an article should be completed in less than one second after a click on a word in any text (full hypertext function);*
- *the display of a multimedia asset should be completed in less than two seconds after a click on its title in a list or within an article;*
- *the results of a full text search should be displayed in less than two seconds after the search has been triggered;*
- *the results of a multi-criteria search, including a period search (defined by beginning and end dates), should be displayed in less than three seconds.*

Testing and acceptance

The requirements document should specify how the product will be tested at the various stages of development. The **final acceptance** tests generally use the requirements specification (in its final approved state) as the "ultimate" benchmark. Intermediate and final testing processes need to be specified in terms of the nature of the tests and the product acceptance criteria.

For business applications and, particularly, for complex information system (IS) projects, specific testing phases may be required and therefore specified by the project owner, such as:

- an "**operational acceptance testing (OAT)**" phase, sometimes called "**operational readiness testing**", intended to determine whether the product meets requirements in a real-life situation, but on a relatively small scale, before it is fully deployed and put into service for all of its target users;

- an "**operational health check (OHC)**", intended to verify that the product meets requirements after being fully deployed and put into unrestricted and regular service.

Delivery medium and installation

The requirements for the medium (or media) on which the product is expected to be delivered, as well as installation options, should be clearly specified.

A software product may be delivered for example on a CD-ROM, a DVD-ROM, as a file (eg a disk image) uploaded to a website, on an SD card, etc.
A website may be delivered as such by the developer or may be provided on a portable medium (such as a CD/DVD-ROM), the content of which will then need to be uploaded to an actual site.

Installation options for a CD/DVD-ROM for example may be: no installation at all (the application runs from the disk); partial installation (only part of the product is installed on the user's hard disk, the remaining part being on the CD/DVD-ROM); full installation.

If a CD/DVD-ROM is to be copy-protected, this requirement should be specified, as well as the copy-protection mechanism if a choice has already been made (which may have an impact on development).

Processes and logistics

The operation of certain applications requires the execution of processes (eg a particular workflow for the completion of specific tasks) and the availability of adequate logistics (eg the storage and shipment of goods ordered from a website). Insofar as they are part of a project, processes and logistics need to be covered in the requirements specification.

> *One of the projects I coordinated as a freelance was the design and development of a start-up company's website, which included an online shop for the purchase of goods (eg leather covers for e-readers). The initial version of the shop was limited to the catalog of goods along with a notice stating that ordering of goods would be made available soon...*
> *Version 2 was supposed to integrate all the functions expected of an online shop: catalog, shopping cart, payment (via PayPal), etc.*
>
> *The development work for the online shop was done by a small firm based in Shanghai, China and it was completed with a relatively small number of development-testing iterations, in full compliance with the requirements.*
> *The shop could however not be opened to the public before the logistics and administrative processes had been put in place. The issues that remained to be resolved were the provision of "Sales Terms & Conditions", the process for sourcing goods, the choice of a logistics partner for the storage and shipment of goods, the processes for order management and customer services.*
> *Those issues had not been addressed by the project owner by the time my contract expired (in Nov. 2007) and were still unresolved the last time I checked (years later): the fully-featured online shop had remained in the state of "vapourware"!*

Documentation and source code

In many cases, a project owner will require software that is developed within the scope of the project to be adequately (more or less extensively) documented. Furthermore, the developer may be required to **transfer ownership** of the source code (and the source code itself) to the project owner upon completion of the project. Such requirements need to be clearly specified.

Those requirements may extend to documentation for administrators, webmasters, technical support personnel and end users.

Training

Any requirement for training, as well as training material, to be delivered by the project implementation entity to the project owner and intended for administrators, support people, users, etc., also needs to be properly specified.

Schedule and milestones

Major milestones relating to the project's **deliverables** should be documented in the requirements specification, possibly as an appendix (which may provide a preliminary project schedule). For example, milestone dates may be given for the following events:

- response (from the project implementation entity) to the requirements specification;
- agreement on requirements by project owner and project implementation entity;
- start of implementation (which requires confirmation of the project implementation entity's availability as of the specified date);
- completion of product design specifications;

- agreement on functional design and user-interface specifications by project owner and project implementation entity;
- start of content creation;
- start of software development (usually done in parallel with content creation);
- delivery of sample data (or data subset) to the software developer;
- first "alpha" version of software (integrating subsets of data and functions);
- acceptance of the alpha version by project owner;
- delivery of full data set to the software developer;
- first "beta" version of software (integrating full data set and all functions);
- acceptance of the beta version by project owner;
- first "release-candidate" (or "master-candidate") version of software;
- results of the testing and acceptance of the final release-candidate version;
- delivery of the final "master" to project owner;
- acceptance of the product by the project owner;
- delivery of software documentation and source code to project owner;
- delivery of training to specified target audiences.

Risks, dependencies and other issues

Any relevant risks, dependencies and other issues that may have been identified at this stage of project preparation should be clearly mentioned in the requirements document, insofar as they need to be taken into account by the project implementation entities, who may actually be encouraged to recommend actions or alternatives in order to reduce the extent and impact of such risks, dependencies and issues.

As an example, here is an excerpt from the "Risks and dependencies" chapter of a requirements specification written by a British publishing company (in 2007):

- *"Web platform flexibility to accommodate and provide required functionality for a wide range of products in the future.*
- *Meeting accessibility requirements for all features of the website.*
- *Conflict with other projects: it will be important to assure that this substantial development does not conflict with any other projects or put undue strain on the resources of [the Developer]."*

Any other relevant information

The above-described components of the requirements specification may of course be complemented with any other information that is judged to be relevant and useful in the context of the particular project concerned, for example environmental considerations.

> See the following site for **information on writing a Software Requirements Specification (SRS)**:

>> techwhirl.com/articles/writingsoftwarerequirementsspecs/

9) Project planning

General remarks

The project charter (covered in chapter 7) incorporates a summary project plan. Before moving on to the actual execution phase of a project, it is necessary to establish a more comprehensive and detailed **project management plan** or **project plan**, which is based on an **in-depth analysis of the requirements specification** and involves **identifying and evaluating the various components of the project**.

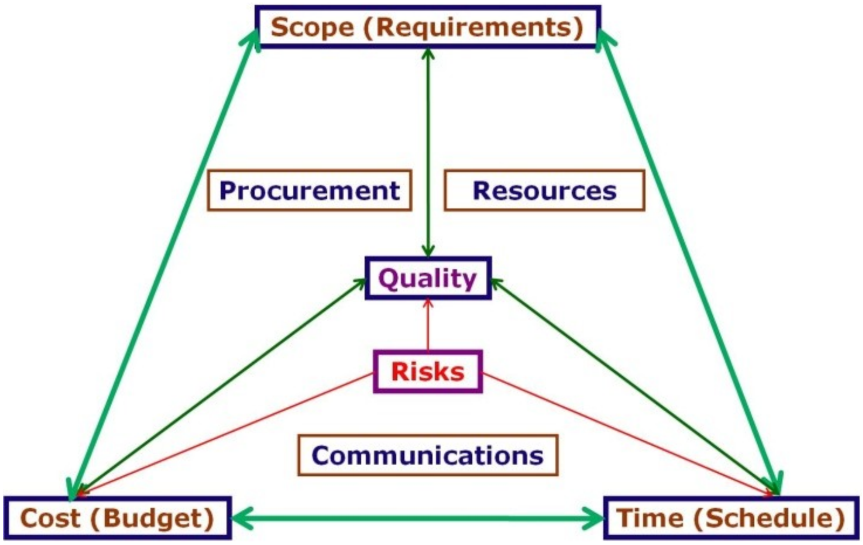
Note that although **the requirements specification is the foundation of the project plan**, it may need to be revised and completed as a result of its in-depth analysis by subject matter experts. That is why preliminary project plans may include the following set of tasks: reviewing, complementing, fine-tuning and agreeing on a revised version of the requirements specification, of the scope of the project and of the project plan itself.

The project plan, which can be considered as the "**road map**" for project execution, defines the **steps** to take and the **resources required** to successfully complete the project, **how long** it will take and **how much** it will cost. It should also include the **identification of risks**, and define **how the project will be executed**, **monitored and controlled, and finally closed**.

The project plan is a **Master plan** incorporating several **subsidiary plans** related to the following **areas of project management** (as listed in the Project Management Institute (PMI) methodology, but not necessarily in the same order).

- **Scope** management
- **Time** management
- **Cost** management
- **Resource** management
- **Procurement** management
- **Quality** management
- **Communications** management
- **Risk** management

Those areas of project management are interrelated and interact with one another, as partially depicted in the following diagram (which also appears at the end of chapter 1).

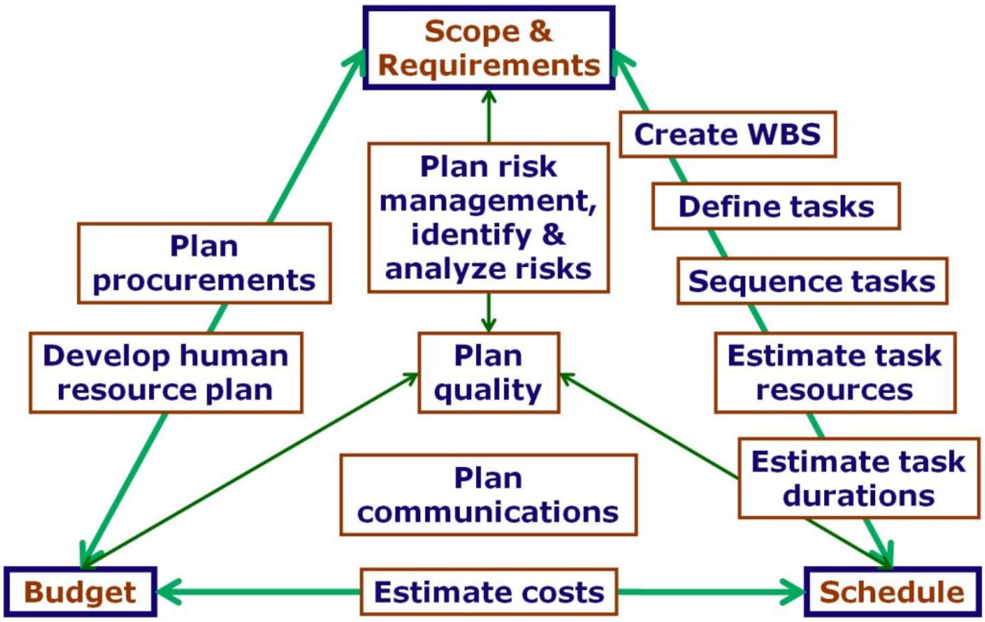


Planning processes: introduction

The **planning processes** related to the project management areas are the following.

- Define scope and Collect requirements.
- Create work breakdown structure (WBS).
- Define tasks.
- Sequence tasks.
- Estimate task resources.
- Estimate task durations.
- Develop schedule.
- Estimate costs.
- Determine budget.
- Develop human resource plan.
- Plan procurements.
- Plan quality.
- Plan communications.
- Plan risk management, identify and analyze risks.

Below is a version of the project management areas diagram which is somewhat rearranged and expanded in order to include some of the above-listed processes.



The first two processes (concerning scope and requirements) have been addressed at the beginning of this chapter as well as in previous chapters of this guide. The following sections of this chapter deal with the other processes. They are presented sequentially, in the above-listed order, but the **processes** and their results are actually very much **interdependent**. For example, the duration of a task depends on the resources applied to it, and if adequate resources are available, work may be divided into tasks that can be executed in parallel. Project planning is therefore necessarily an **iterative process**.

Create the work breakdown structure (WBS)

Creating the "**work breakdown structure** (**WBS**)" consists in **dividing the project** and related work **into "manageable" components**.

The WBS provides an **overview** of the total scope of the project and of its organization. The **higher levels** of the structure, which is where to start when creating a WBS, may correspond to **phases** of the project, to **subprojects** or to project **work areas**.
The **lowest level** may consist of **tasks** (sometimes called activities) or **groups of tasks** called "**work packages (WPs)"** or a combination of tasks and WPs.
The level of detail of WPs that needs to be shown in a WBS varies with the size and complexity of a project.

It is important to ensure that the WBS covers 100% (no less, no more) of the work to be done to produce all of the deliverables, It should also take into account project management work, but not necessarily documented at the lowest level of detail.
This is the so-called "**100% rule**", which also applies to the set of tasks within any WP.

The **WBS** may be represented by a **text outline** (comparable to a table of contents) and/or by a **diagram** (generally a tree structure). The diagram may be less detailed than the text version of the WBS. As a starting point, it is useful to draw a **simple WBS diagram** (preferably on a **single page** for the sake of readability) to provide a **global view of the work to be done** in order to complete the project.

The WBS may be supplemented with a **WBS dictionary**, the purpose of which is to document details of the work packages (WPs). Each WP entry typically includes an identifier, a brief description of the work to be performed, and the organization, team or individual responsible for the work.

The WBS is a fundamental project management tool. It serves as a reference for many processes such as scheduling and costing, resource assignment and risk assessment. It provides a framework for project supervision and is useful for communication purposes.

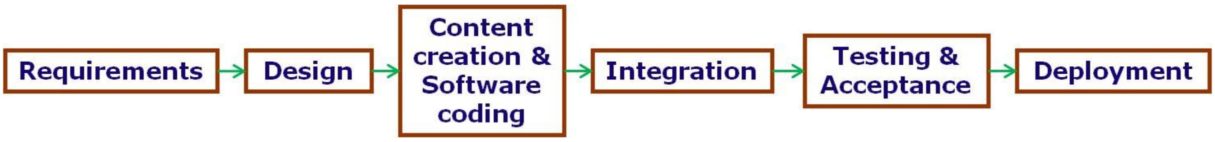
As project work advances, changes to the WBS may be required. The initial WBS should however be retained as a **baseline for change control**.

The **requirements specification** is the major **source of information** for the creation of the WBS. A summary WBS may actually have been created in the process of conducting a feasibility study, preparing the business case and the project charter.

In order to develop the WBS, the PM may need **help from experts** in the various project areas. Such experts may be people in the organization who have already been assigned to the project. Obviously, the PM and other people participating in the effort need to use their **imagination** to build an appropriate WBS for the project!
The WBS of **previous, similar projects** may be helpful as a starting point.

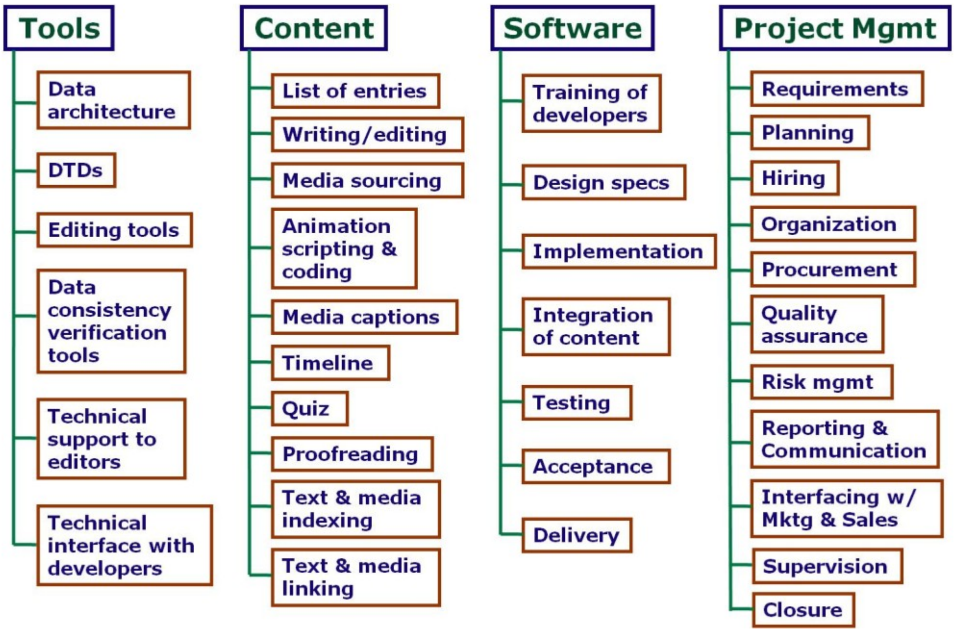
Note that projects resulting in an information system, a software application or a website generally involve the **major phases** shown in the diagram below. The corresponding subprojects or work packages should therefore appear in some form in the WBS of such projects. Furthermore, the particular **software development model or methodology** that may be chosen has a direct impact on a project's WBS (as presented in chapter 12).

Also note that **content creation and software coding** (implementation) can generally be executed **in parallel**.



The initial EHM project was broken down into subprojects and work packages as follows (text outline followed by a diagram version of the WBS):

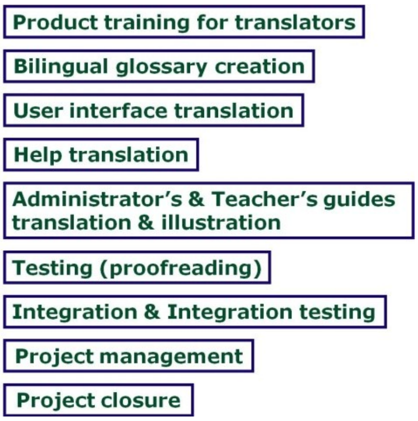
- *<u>"Tools" subproject</u> (person in charge: Director of Data Engineering):*
    - *Design of data architecture and structure*
    - *Development of DTDs (encyclopedia and dictionary)*
    - *Development of editing tools (for data and metadata)*
    - *Development of data consistency verification tools*
    - *Technical support to editorial team*
    - *Technical interface with developers*
- *<u>"Content" subproject</u> (person in charge: Editorial Director):*
    - *Creation of the list of entries*
    - *Text writing/editing*
    - *Sourcing of multimedia assets (photos, drawings, audiovisuals, etc.)*
    - *Scripting and development of animations*
    - *Writing media captions*
    - *Development of the Timeline*
    - *Development of the Quiz*
    - *Text proofreading*
    - *Indexing of texts and multimedia assets*
    - *Linking of media to texts, and of texts to texts*
- *<u>"Software" subproject</u> (person in charge: Project Director, with the assistance of the Director of Data Engineering and the Editorial Director):*
    - *Training of developers on data structure*
    - *Cooperation with developers for the writing of design specifications*
    - *Implementation of the software by the subcontractor*
    - *Integration of content*
    - *Testing, acceptance and delivery of the finished product*
- *<u>"Project Management" subproject</u> (person in charge: Project Director):*
    - *Requirements specification*
    - *Planning*
    - *Hiring*
    - *Organization*
    - *Procurement (incl. choice of subcontractor for software development)*
    - *Quality assurance*
    - *Risk management*
    - *Reporting and communication*
    - *Interfacing with Marketing & Sales*
    - *Supervision, monitoring and control*
    - *Closure*



> *The "**Implementation**" work package that appears under "Software" in the previous example of the EHM project WBS was a **major subproject** in itself. This subproject obviously required a specific WBS, which was created by the development contractor for the purpose of detailed development planning and management.*

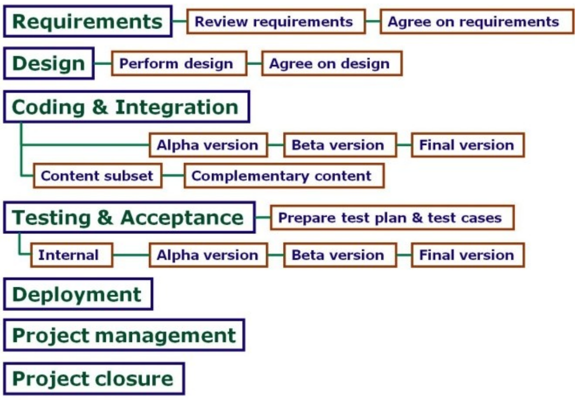
Note that the "**Project management**" work package **generally does not need to be described at a low level of detail** (the details in the previous example were given for purely pedagogical purposes as a reminder of what project management involves!). Indeed, apart from "Requirements" and "Project closure", which generally involve more resources than just the PM, the "Project management" tasks may be featured in the WBS as a single item (work package) that implicitly includes all types of activity and work required to manage the project.
It may also be split into "usual work" and "exceptional work", the latter concerning tasks that may need to be evaluated explicitly in the project plan (eg in terms of duration), because of their impact on the overall schedule (eg "Hiring" and "Procurement").
"Project management" may also be divided into phases, for example in the case where the PM is expected to work full-time on the project in some of its phases and only part-time in other phases.

As another example, here is the simple high-level WBS of a product localization project.

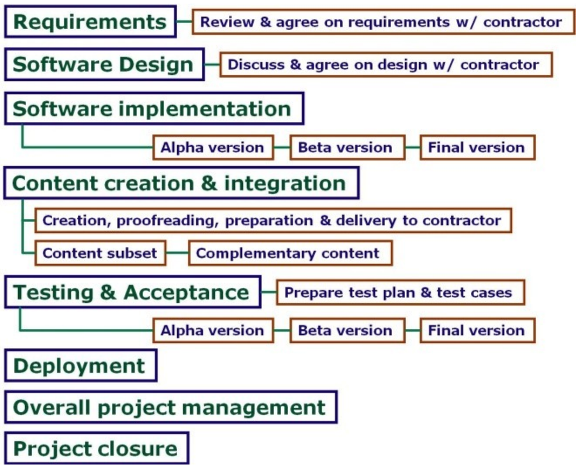

Product training for translators
Bilingual glossary creation
User interface translation
Help translation
Administrator's & Teacher's guides translation & illustration
Testing (proofreading)
Integration & Integration testing
Project management
Project closure

Finally, here are two WBS diagrams of a "hypothetical" project named **EXONE** (project EXONE is used as an example on several occasions in this guide). The WBS on the left covers the work to be done by a software development contractor for a client, the WBS on the right represents the client's view of the work to be done by both the client and the contractor. The client's WBS has been derived from the contractor's WBS, which was submitted to the client, along with a tentative project plan, after the contractor had analyzed the client's requirement specification.

Contractor's WBS



Client's WBS


Requirements
Review & agree on requirements w/ contractor
Software Design
Discuss & agree on design w/ contractor
Software implementation
Alpha version
Beta version
Final version
Content creation & integration
Creation, proofreading, preparation & delivery to contractor
Content subset
Complementary content
Testing & Acceptance
Prepare test plan & test cases
Alpha version
Beta version
Final version
Deployment
Overall project management
Project closure

Define and sequence tasks

General remarks

A complete work breakdown structure (WBS) is divided into **work packages (WPs)** which include more **detailed work components** called **tasks** (or activities), for which estimates can be made in terms of resources required, duration and, as a result, cost.

The process of defining tasks will eventually result in a **list** that covers the **whole set of WPs/tasks** that are **necessary to complete the project**. The WPs/tasks should be presented in a hierarchical fashion, eg a list of **WPs broken down into a specific list of subordinate tasks** whenever such detail is necessary for any given WP.
The term "**WBS**" is often used in project management software applications to refer to the **complete hierarchical list of tasks of a project**.

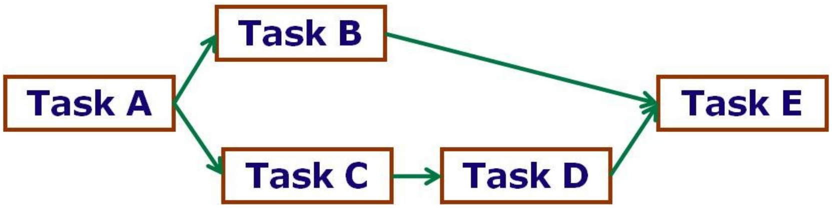
Tasks may be given **attributes**: identification code, description, resource requirements, duration, assumptions, dependencies...
Not all of the information relating to each task is necessarily available at the beginning of the task definition process, so the set of task attributes will become more complete and detailed as the planning effort moves forward.

The process of identifying the set of tasks required to complete the project is usually made easier by first **drawing** a so-called "**network diagram**", which is generally more detailed than the WBS diagram and shows how tasks should be sequenced, as explained below.

Network diagram

The sequence of tasks in a project may be represented by a **network diagram** where **nodes** represent **tasks** (or work packages), and **arrows** show the **logical relationships** between them.
(Such relationships are generally **chronological**.)

Here is a partial network diagram showing the relationships between five tasks in an imaginary project:



The above diagram provides the following information:

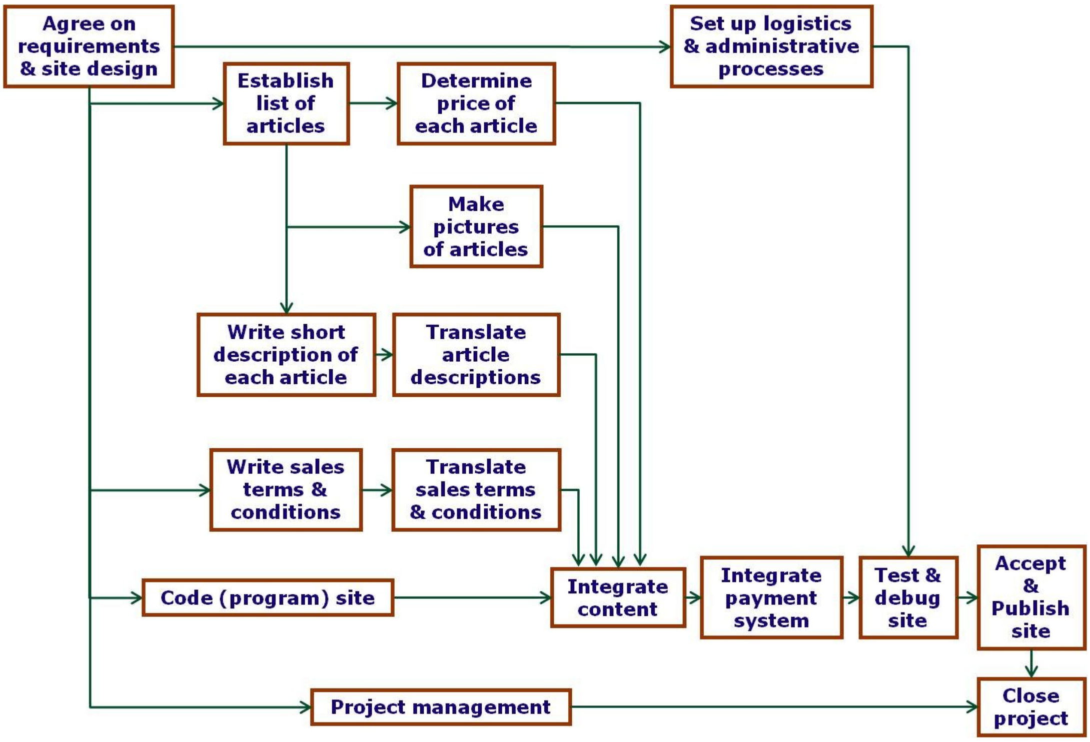
- tasks B and C, which are executed in parallel, cannot start until task A has been completed,
- task D cannot start until task C has been completed,
- task E cannot start until tasks B and D have been completed.

As mentioned above, it is useful to draw a **simple network diagram** (preferably on a **single page** for the sake of readability) to provide a **global view of the sequence of tasks** in the project.

Additional diagrams may be necessary to show the sequence of tasks within the various work packages (WPs) at a **more detailed level**.

A few examples of network diagrams are given hereafter.

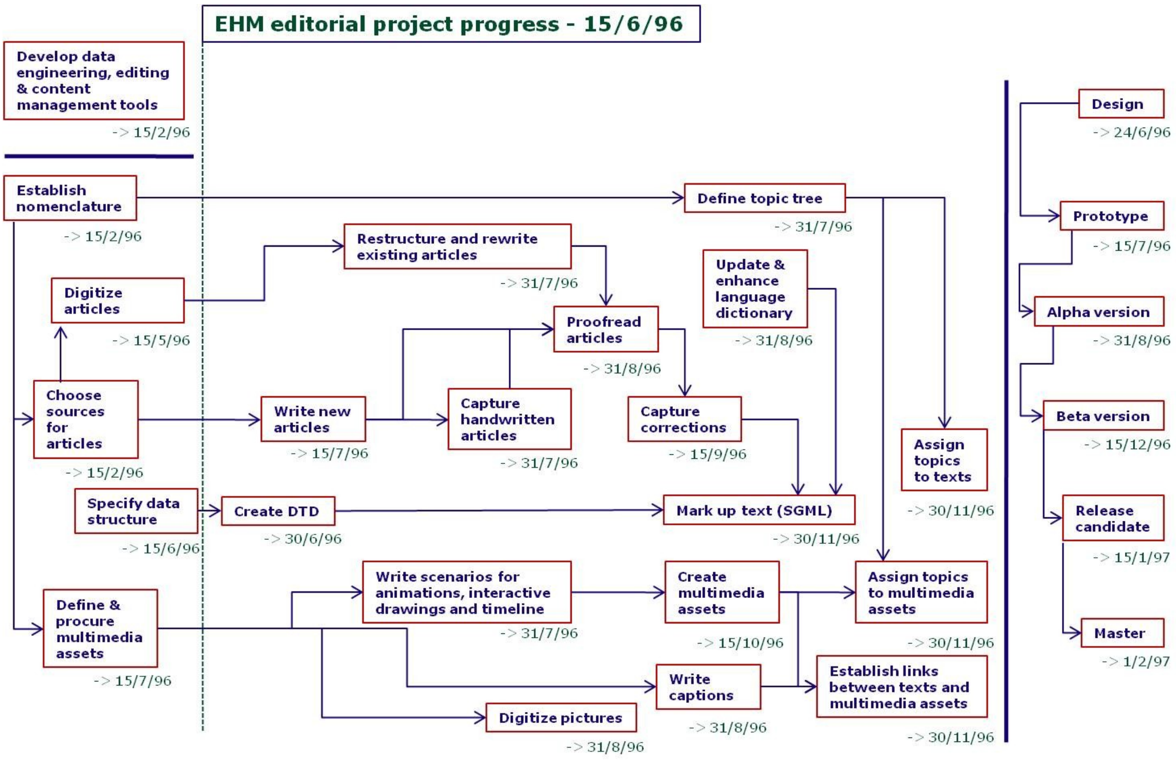
Here is a simplified diagram showing the sequence of WPs/tasks in a project for the creation of an online store.



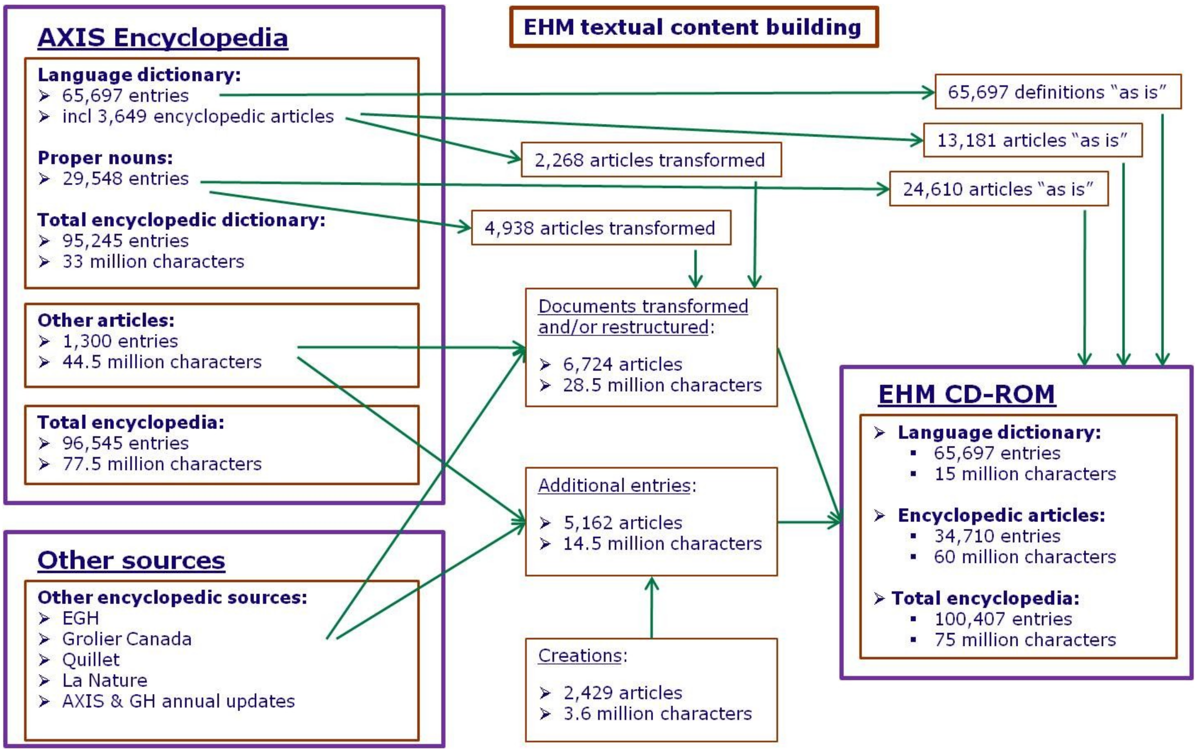
Agree on requirements & site design
Set up logistics & administrative processes
Establish list of articles
Determine price of each article
Make pictures of articles
Write short description of each article
Translate article descriptions
Write sales terms & conditions
Translate sales terms & conditions
Code (program) site
Integrate content
Integrate payment system
Test & debug site
Accept & Publish site
Project management
Close project

Even though the detailed scheduling process may not be complete at this stage of planning, schedule information (eg major milestones) or any other **available relevant information** (eg human resources) may be featured in a summary network diagram in order to make it more informative.

> *The following example is a one-page network diagram that provides a global view of the progress of the initial EHM project at a fairly advanced stage (some tasks had already been completed). The focus is on the "Content" (editorial) WPs. "Tools" and "Software" WPs are summarized respectively in the top left box and on the right.*

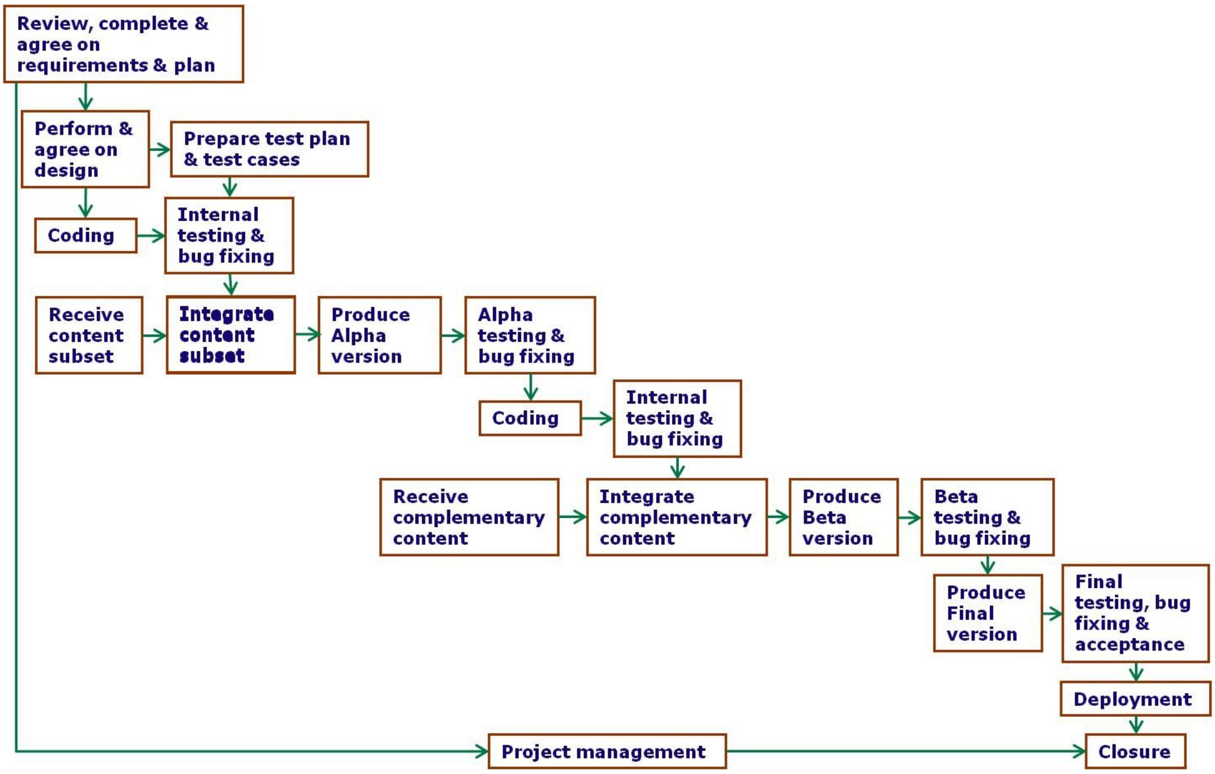


The following diagram shows how the textual content of the EHM was to be sourced or created, and finally assembled. It was used as a summary view of background information required for establishing the editorial part of the previous network diagram.

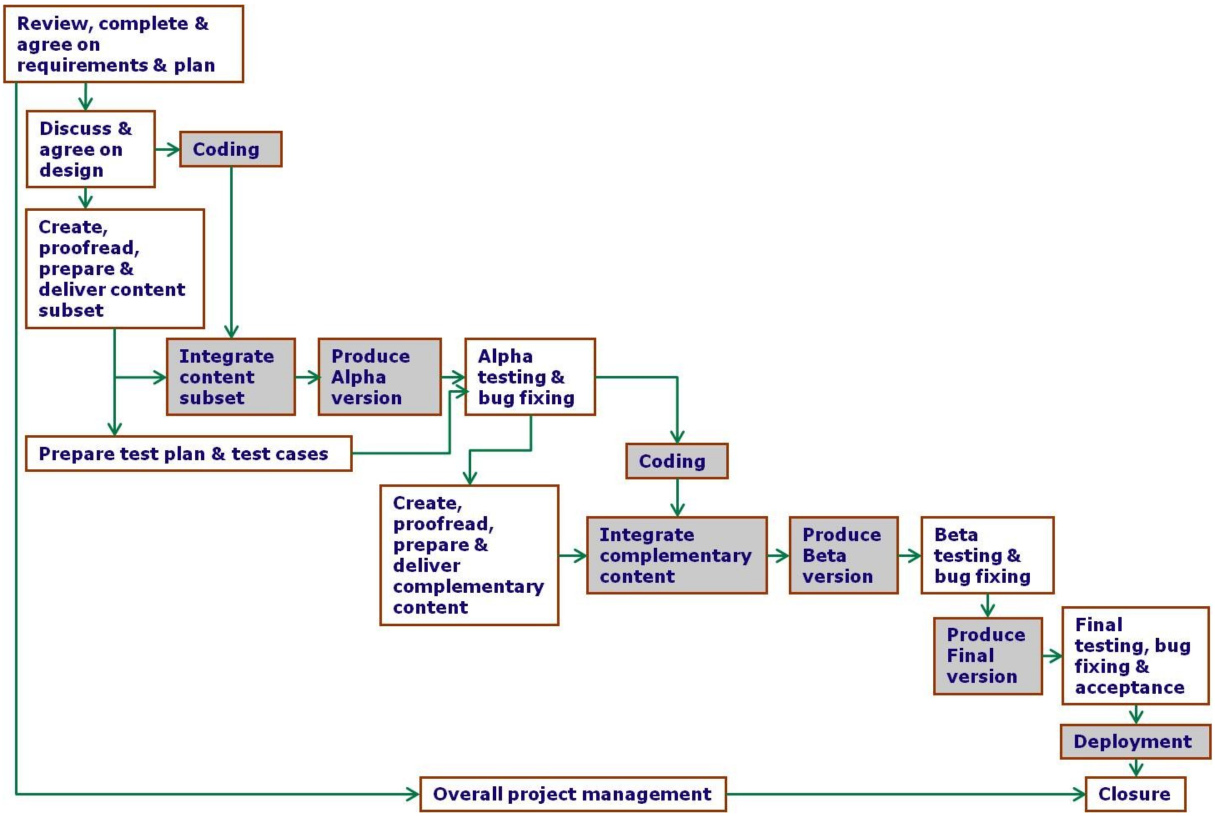


Other example: the following two summary diagrams show the sequence of WPs/tasks in project EXONE, first from the contractor's point of view then from the client's point of view (refer to this project's WBS diagrams on a previous page).

Contractor's network diagram


Review, complete & agree on requirements & plan
Perform & agree on design
Prepare test plan & test cases
Coding
Internal testing & bug fixing
Receive content subset
Integrate content subset
Produce Alpha version
Alpha testing & bug fixing
Coding
Internal testing & bug fixing
Receive complementary content
Integrate complementary content
Produce Beta version
Beta testing & bug fixing
Produce Final version
Final testing, bug fixing & acceptance
Deployment
Project management
Closure

Client's network diagram



Review, complete & agree on requirements & plan
Discuss & agree on design
Coding
Create, proofread, prepare & deliver content subset
Integrate content subset
Produce Alpha version
Alpha testing & bug fixing
Prepare test plan & test cases
Coding
Create, proofread, prepare & deliver complementary content
Integrate complementary content
Produce Beta version
Beta testing & bug fixing
Produce Final version
Final testing, bug fixing & acceptance
Deployment
Overall project management
Closure

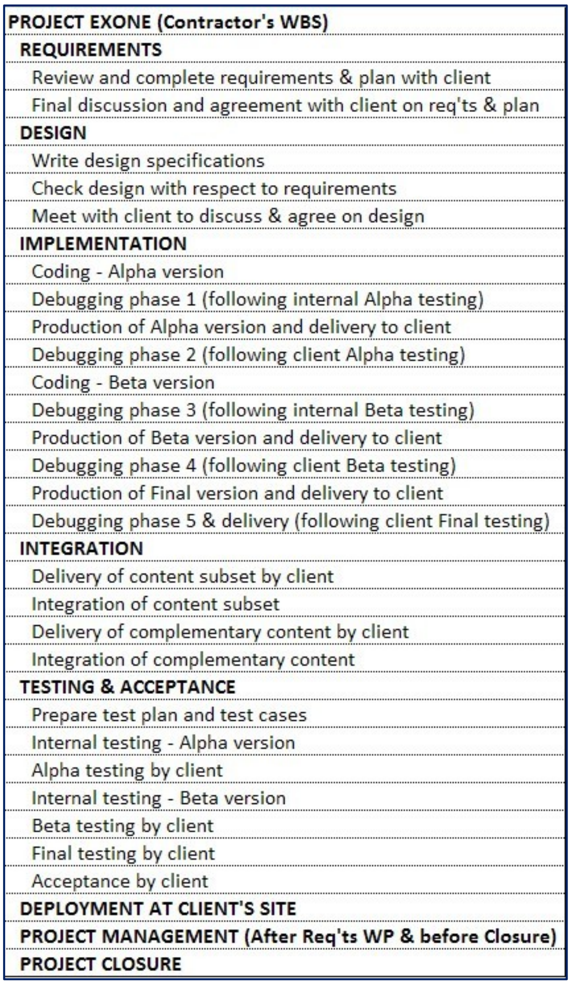
In the above diagram the boxes with a grey background represent WPs/tasks to be performed exclusively by the contractor; the other WPs/tasks concern either work to be done exclusively by the client or work involving both the client and the contractor. Including work to be done by the contractor in the client's network diagram provides the client with an **overall view of the project**.

Note that the expression "**Overall project management**" is used in order to distinguish project management on the client's side from project management on the contractor's side. The project manager working for the client (project owner) has **overall responsibility for the whole project**, whereas the project manager working for the contractor is only responsible for work done by the contractor.

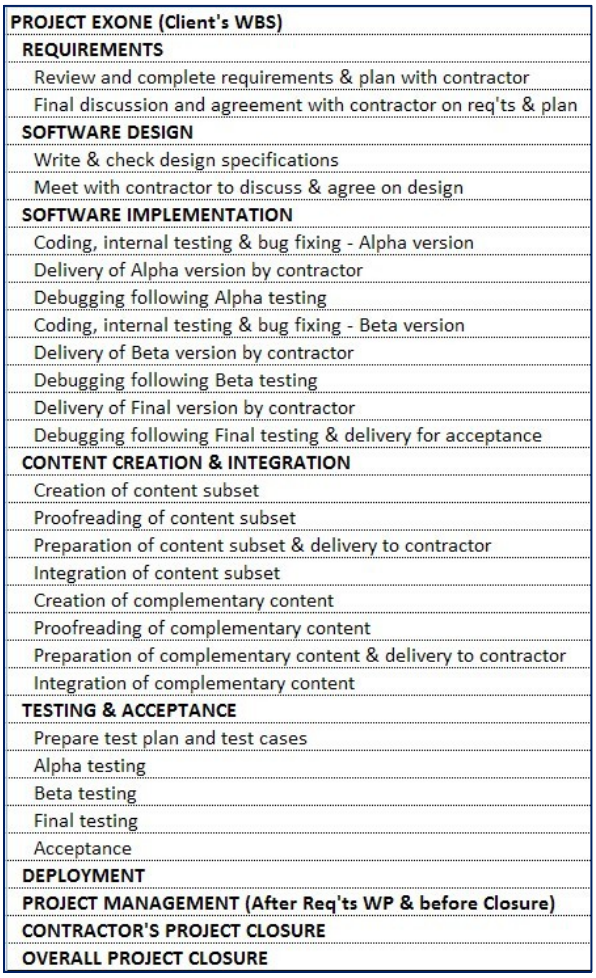
As demonstrated by the previous examples, a **network diagram** can be **created "by hand"**, possibly with a software application that incorporates drawing tools. It can also be produced automatically from a sequenced list of tasks by a project management application such as Microsoft Project or ProjectLibre, as explained further on, but such applications do not necessarily provide an **easy-to-read single-page overview**.

Task list

The two illustrations below are **structured task lists** for project EXONE, respectively from the contractor's standpoint and from the client's standpoint, derived from the network diagrams of the project, shown on previous pages.



PROJECT EXONE (Contractor's WBS)
REQUIREMENTS
Review and complete requirements & plan with client
Final discussion and agreement with client on req'ts & plan
DESIGN
Write design specifications
Check design with respect to requirements
Meet with client to discuss & agree on design
IMPLEMENTATION
Coding - Alpha version
Debugging phase 1 (following internal Alpha testing)
Production of Alpha version and delivery to client
Debugging phase 2 (following client Alpha testing)
Coding - Beta version
Debugging phase 3 (following internal Beta testing)
Production of Beta version and delivery to client
Debugging phase 4 (following client Beta testing)
Production of Final version and delivery to client
Debugging phase 5 & delivery (following client Final testing)
INTEGRATION
Delivery of content subset by client
Integration of content subset
Delivery of complementary content by client
Integration of complementary content
TESTING & ACCEPTANCE
Prepare test plan and test cases
Internal testing - Alpha version
Alpha testing by client
Internal testing - Beta version
Beta testing by client
Final testing by client
Acceptance by client
DEPLOYMENT AT CLIENT'S SITE
PROJECT MANAGEMENT (After Req'ts WP & before Closure)
PROJECT CLOSURE



PROJECT EXONE (Client's WBS)
REQUIREMENTS
Review and complete requirements & plan with contractor
Final discussion and agreement with contractor on req'ts & plan
SOFTWARE DESIGN
Write & check design specifications
Meet with contractor to discuss & agree on design
SOFTWARE IMPLEMENTATION
Coding, internal testing & bug fixing - Alpha version
Delivery of Alpha version by contractor
Debugging following Alpha testing
Coding, internal testing & bug fixing - Beta version
Delivery of Beta version by contractor
Debugging following Beta testing
Delivery of Final version by contractor
Debugging following Final testing & delivery for acceptance
CONTENT CREATION & INTEGRATION
Creation of content subset
Proofreading of content subset
Preparation of content subset & delivery to contractor
Integration of content subset
Creation of complementary content
Proofreading of complementary content
Preparation of complementary content & delivery to contractor
Integration of complementary content
TESTING & ACCEPTANCE
Prepare test plan and test cases
Alpha testing
Beta testing
Final testing
Acceptance
DEPLOYMENT
PROJECT MANAGEMENT (After Req'ts WP & before Closure)
CONTRACTOR'S PROJECT CLOSURE
OVERALL PROJECT CLOSURE

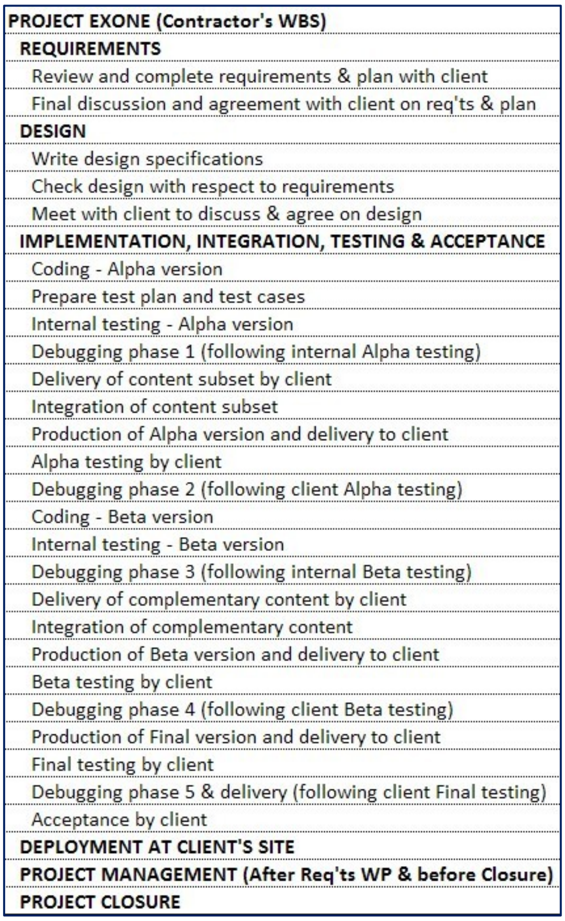
A project's task list needs to be **exhaustive**, so it is generally more detailed than the diagram version of the WBS and may be more detailed than the "hand-drawn" network diagram.

Although it is not mandatory to do so, the name of the project, in this case "**PROJECT EXONE**", is featured at the **highest level of the task list** on the first line of the above tables, so that **consolidated information at project level** can be viewed easily, for example the total duration, the overall schedule and the total cost of the project, which are automatically calculated by project management applications, as will be explained and illustrated further on in this chapter.

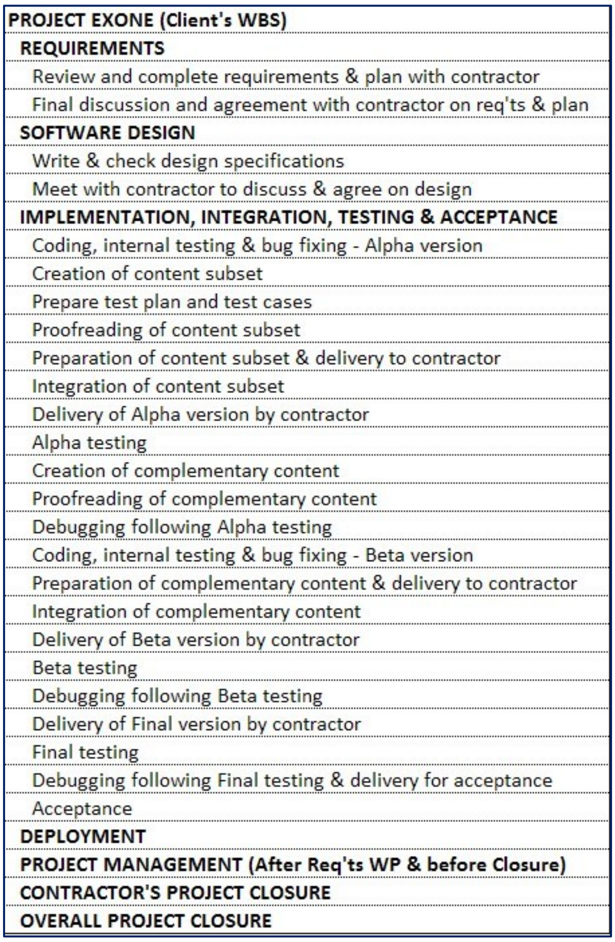
Note that the client's task list incorporates some of the tasks that appear in the contractor's task list, with an appropriate level of detail, in order to provide the client with an overall view of the project, including those contractor's WPs/tasks on which the client's WPs/tasks are dependent.

Also note that the client's list features two "Project closure" entries, the first one corresponding to the closure of the contractor's project, the second one concerning the closure of the project from the client's standpoint.

The set of tasks shown on the previous page may be presented with some of the work packages grouped together, and in a somewhat different order, as shown below.



PROJECT EXONE (Contractor's WBS)
REQUIREMENTS
Review and complete requirements & plan with client
Final discussion and agreement with client on req'ts & plan
DESIGN
Write design specifications
Check design with respect to requirements
Meet with client to discuss & agree on design
IMPLEMENTATION, INTEGRATION, TESTING & ACCEPTANCE
Coding - Alpha version
Prepare test plan and test cases
Internal testing - Alpha version
Debugging phase 1 (following internal Alpha testing)
Delivery of content subset by client
Integration of content subset
Production of Alpha version and delivery to client
Alpha testing by client
Debugging phase 2 (following client Alpha testing)
Coding - Beta version
Internal testing - Beta version
Debugging phase 3 (following internal Beta testing)
Delivery of complementary content by client
Integration of complementary content
Production of Beta version and delivery to client
Beta testing by client
Debugging phase 4 (following client Beta testing)
Production of Final version and delivery to client
Final testing by client
Debugging phase 5 & delivery (following client Final testing)
Acceptance by client
DEPLOYMENT AT CLIENT'S SITE
PROJECT MANAGEMENT (After Req'ts WP & before Closure)
PROJECT CLOSURE



PROJECT EXONE (Client's WBS)
REQUIREMENTS
Review and complete requirements & plan with contractor
Final discussion and agreement with contractor on req'ts & plan
SOFTWARE DESIGN
Write & check design specifications
Meet with contractor to discuss & agree on design
IMPLEMENTATION, INTEGRATION, TESTING & ACCEPTANCE
Coding, internal testing & bug fixing - Alpha version
Creation of content subset
Prepare test plan and test cases
Proofreading of content subset
Preparation of content subset & delivery to contractor
Integration of content subset
Delivery of Alpha version by contractor
Alpha testing
Creation of complementary content
Proofreading of complementary content
Debugging following Alpha testing
Coding, internal testing & bug fixing - Beta version
Preparation of complementary content & delivery to contractor
Integration of complementary content
Delivery of Beta version by contractor
Beta testing
Debugging following Beta testing
Delivery of Final version by contractor
Final testing
Debugging following Final testing & delivery for acceptance
Acceptance
DEPLOYMENT
PROJECT MANAGEMENT (After Req'ts WP & before Closure)
CONTRACTOR'S PROJECT CLOSURE
OVERALL PROJECT CLOSURE

That second example of task list shows **tasks in the order in which they should be executed**. This straightforward ordering of tasks is **recommended when using ProjectLibre** (an alternative to Microsoft Project), which in the case of more sophisticated ordering may flag certain task sequences as erroneous although they may actually be correct.

As compared to ProjectLibre, **Microsoft Project is a more powerful application** and allows more flexibility in ordering tasks.

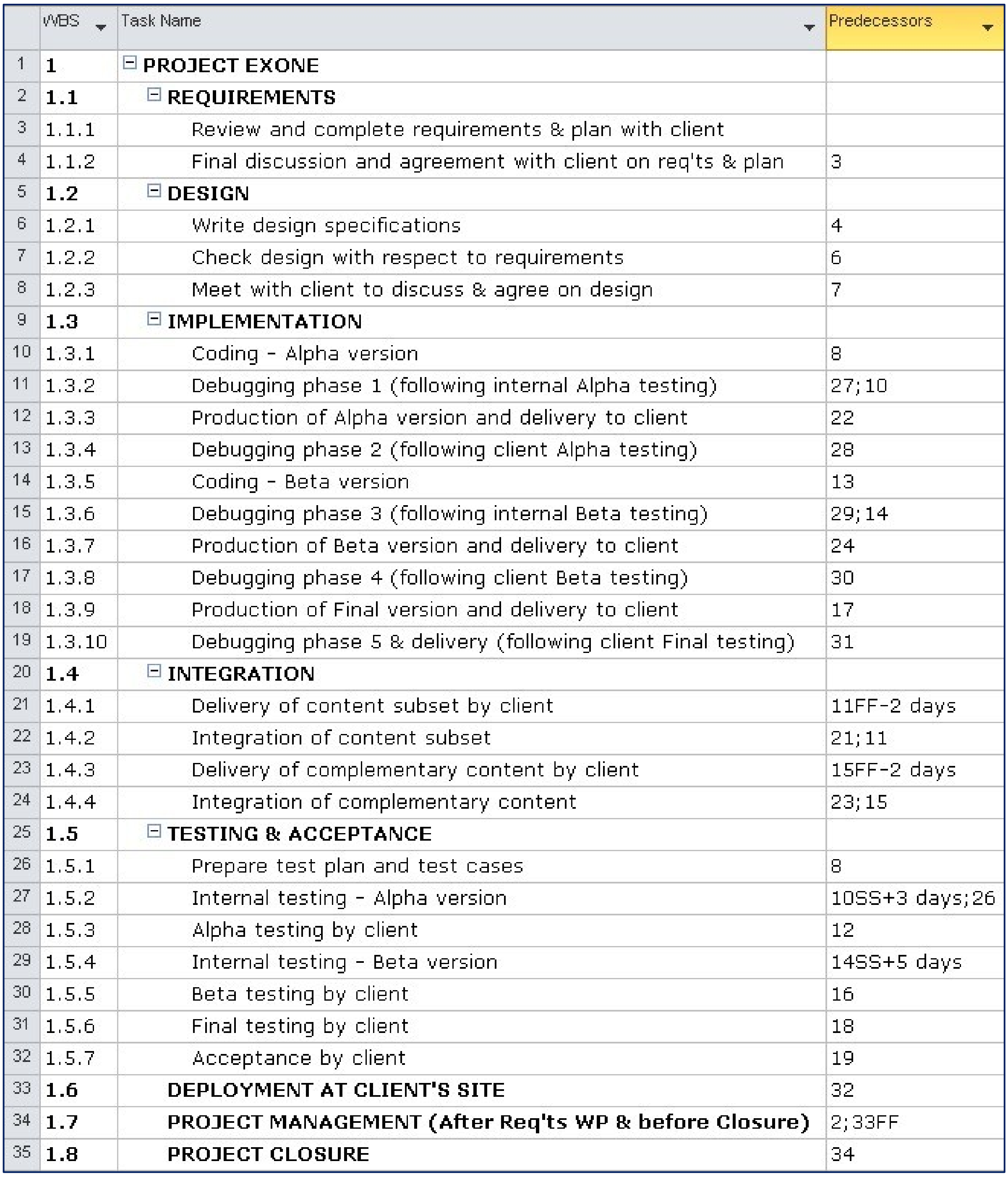
Note that producing a project plan does not necessarily require the use of specialized project management software. A multi-purpose tool such as **a spreadsheet application** (eg Microsoft Excel) **can be used for projects that are not too complex** in terms of WBS and task sequencing.

Specialized applications do however make project planning (and monitoring) easier thanks to **built-in tools** such as automatic scheduling and cost calculation, and to a **variety of "views"** which provide useful information such as resource usage and critical paths, as explained further on in this chapter.

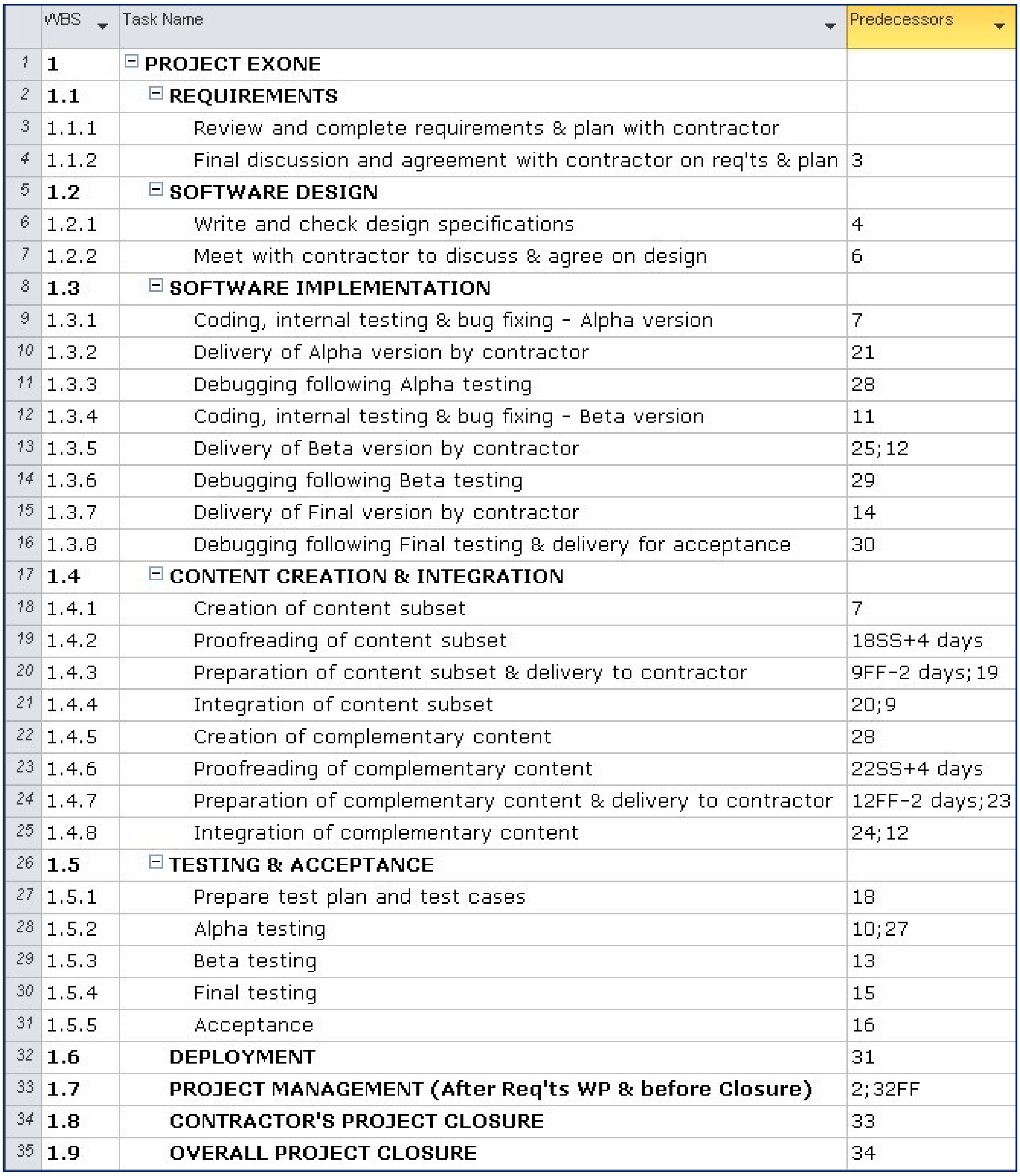
Note that most of the screenshots concerning the EXONE project featured in subsequent sections of this guide are illustrations produced with Microsoft Project, not with ProjectLibre. Other screenshots represent documents produced with Microsoft Excel.

Task sequencing

The so-called “**Gantt view**” of a project plan built with MS Project (or ProjectLibre) consists of a table showing in the "Task name" column the comprehensive list of project tasks, hierarchically organized in work packages, and additional information in other columns.
One of those additional columns, labelled “**Predecessors**”, is where the **dependencies between tasks** are set, using the **line numbers** in the leftmost column of the table, as shown in the following illustrations for project EXONE, first for the contractor’s plan then for the client’s plan.



	WBS	Task Name	Predecessors
1	**1**	⊟ **PROJECT EXONE**	
2	**1.1**	⊟ **REQUIREMENTS**	
3	1.1.1	Review and complete requirements & plan with client	
4	1.1.2	Final discussion and agreement with client on req'ts & plan	3
5	**1.2**	⊟ **DESIGN**	
6	1.2.1	Write design specifications	4
7	1.2.2	Check design with respect to requirements	6
8	1.2.3	Meet with client to discuss & agree on design	7
9	**1.3**	⊟ **IMPLEMENTATION**	
10	1.3.1	Coding - Alpha version	8
11	1.3.2	Debugging phase 1 (following internal Alpha testing)	27;10
12	1.3.3	Production of Alpha version and delivery to client	22
13	1.3.4	Debugging phase 2 (following client Alpha testing)	28
14	1.3.5	Coding - Beta version	13
15	1.3.6	Debugging phase 3 (following internal Beta testing)	29;14
16	1.3.7	Production of Beta version and delivery to client	24
17	1.3.8	Debugging phase 4 (following client Beta testing)	30
18	1.3.9	Production of Final version and delivery to client	17
19	1.3.10	Debugging phase 5 & delivery (following client Final testing)	31
20	**1.4**	⊟ **INTEGRATION**	
21	1.4.1	Delivery of content subset by client	11FF-2 days
22	1.4.2	Integration of content subset	21;11
23	1.4.3	Delivery of complementary content by client	15FF-2 days
24	1.4.4	Integration of complementary content	23;15
25	**1.5**	⊟ **TESTING & ACCEPTANCE**	
26	1.5.1	Prepare test plan and test cases	8
27	1.5.2	Internal testing - Alpha version	10SS+3 days;26
28	1.5.3	Alpha testing by client	12
29	1.5.4	Internal testing - Beta version	14SS+5 days
30	1.5.5	Beta testing by client	16
31	1.5.6	Final testing by client	18
32	1.5.7	Acceptance by client	19
33	**1.6**	**DEPLOYMENT AT CLIENT'S SITE**	32
34	**1.7**	**PROJECT MANAGEMENT (After Req'ts WP & before Closure)**	2;33FF
35	**1.8**	**PROJECT CLOSURE**	34



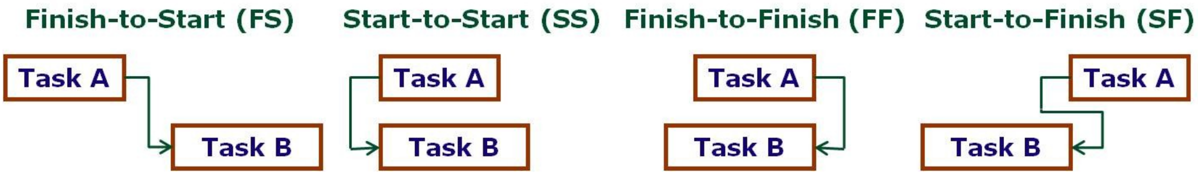
	WBS	Task Name	Predecessors
1	**1**	⊟ **PROJECT EXONE**	
2	**1.1**	⊟ **REQUIREMENTS**	
3	1.1.1	Review and complete requirements & plan with contractor	
4	1.1.2	Final discussion and agreement with contractor on req'ts & plan	3
5	**1.2**	⊟ **SOFTWARE DESIGN**	
6	1.2.1	Write and check design specifications	4
7	1.2.2	Meet with contractor to discuss & agree on design	6
8	**1.3**	⊟ **SOFTWARE IMPLEMENTATION**	
9	1.3.1	Coding, internal testing & bug fixing - Alpha version	7
10	1.3.2	Delivery of Alpha version by contractor	21
11	1.3.3	Debugging following Alpha testing	28
12	1.3.4	Coding, internal testing & bug fixing - Beta version	11
13	1.3.5	Delivery of Beta version by contractor	25;12
14	1.3.6	Debugging following Beta testing	29
15	1.3.7	Delivery of Final version by contractor	14
16	1.3.8	Debugging following Final testing & delivery for acceptance	30
17	**1.4**	⊟ **CONTENT CREATION & INTEGRATION**	
18	1.4.1	Creation of content subset	7
19	1.4.2	Proofreading of content subset	18SS+4 days
20	1.4.3	Preparation of content subset & delivery to contractor	9FF-2 days;19
21	1.4.4	Integration of content subset	20;9
22	1.4.5	Creation of complementary content	28
23	1.4.6	Proofreading of complementary content	22SS+4 days
24	1.4.7	Preparation of complementary content & delivery to contractor	12FF-2 days;23
25	1.4.8	Integration of complementary content	24;12
26	**1.5**	⊟ **TESTING & ACCEPTANCE**	
27	1.5.1	Prepare test plan and test cases	18
28	1.5.2	Alpha testing	10;27
29	1.5.3	Beta testing	13
30	1.5.4	Final testing	15
31	1.5.5	Acceptance	16
32	**1.6**	**DEPLOYMENT**	31
33	**1.7**	**PROJECT MANAGEMENT (After Req'ts WP & before Closure)**	2;32FF
34	**1.8**	**CONTRACTOR'S PROJECT CLOSURE**	33
35	**1.9**	**OVERALL PROJECT CLOSURE**	34

Information entered in the Predecessors column determines the sequencing of tasks (as shown in the “hand-drawn” network diagram but sometimes with a greater level of detail, and, possibly, with slight sequencing differences).

Note that **"Predecessor" information is always entered at the lowest level of the hierarchical list**, namely at task level, or WP level if the WP has no explicit subordinate tasks.

Precedence Diagramming Method

Task sequencing is performed by applying (sometimes unconsciously!) the so-called "Precedence Diagramming Method (**PDM**)", which includes four types of dependencies or (chrono)logical relationships between tasks, as illustrated below.



- **Finish-to-Start (FS)**: task B cannot start until task A has been completed.

FS is the **most commonly used type of relationship**. It is set by **default** with MS Project and ProjectLibre.
For example, in the contractor's tabular view of project EXONE, "Write design specifications" has "Final discussion and agreement with client on req'ts & plan" as a predecessor, with an implicit **FS** relationship, meaning that design cannot start before agreement on the requirements has been reached with the client.

- **Start-to-Start (SS)**: the start of task B is dependent on the start of task A.

For example, the translation of web pages from one language to another cannot start before writing pages in the original language has started. The translation task may however start some time after writing has begun but without waiting for all pages to be written and proofread. Likewise, as shown in the client's tabular view of project EXONE, proofreading of content is to start some time after the start of content creation. In such cases there is an "**SS**" relationship between the tasks.
Other example: debugging can be started as soon as a bug has been reported in the "bug tracking system" by the testers, without waiting for all bugs to be documented.
In both examples, task B will be completed some time after the end of task A.

- **Finish-to-Finish (FF)**: completion of task B is dependent on completion of task A.

For example, in project EXONE, the "Project management" WP has the "Deployment" WP as a predecessor with an "**FF**" relationship, which means that both WPs are scheduled to be completed at the same time, both before project closure. Note that this "FF" relationship was not shown in the EXONE network diagrams featured a few pages back, but was introduced for pedagogical purposes in the detailed tabular view of task sequencing.

- **Start-to-Finish (SF)**: task B cannot be completed until task A has started.

SF is the least commonly used relationship, the other three generally being sufficient to describe the relationships between tasks in a project, with the possible addition of so-called "lags" and "leads", which are explained below.

Lags and leads

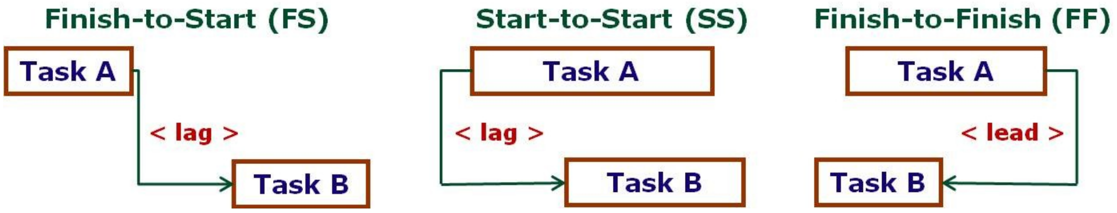
A **lag** is a situation where a **delay** is required between the start of a task and the start of its successor, or between the completion of a task and the completion of its successor.

Using the contractor's plan for project EXONE as an example, assuming that internal testing of the Alpha version (task 1.5.2 – line 27) is expected to begin three days after the start of coding (task 1.3.1 – line 10), then the relationship between these tasks should be defined as "10SS + 3 days".

A **lead** is a situation where a successor task can or must be started or completed sometime **before** its predecessor is started or completed.

For example, in the contractor's plan for project EXONE, it is assumed that the delivery of the content subset by the client (task 1.4.1 – line 21) is required by the contractor, as a safety margin, two days before completion of the first debugging phase (task 1.3.2 – line 11). The relationship between these tasks is therefore defined as "11FF - 2 days". Likewise, in the client's plan, the scheduling constraint imposed by the contractor (and accepted by the client) for the delivery of the content subset (task 1.4.3 – line 20) is featured as a "9FF - 2 days" relationship, where line 9 corresponds to "Coding, internal testing & bug fixing - Alpha version".

Lags are often applied to FS or SS relationships and leads are often used with FF relationships, as shown in the following illustration (SF relationship deliberately omitted, to keep things simple…).



The **lengths of lags and leads** set at an early stage of the planning process are often rough estimates which **need to be adjusted** after task durations have been established.

Estimate task resources

An in-depth analysis of the tasks to be performed should lead to the identification and evaluation of the **types and quantities of resources** required for each task. Resources include people, facilities, materials, equipment, supplies, services, etc.

Defining and estimating resources required for a project is a difficult exercise, and it is obviously easier for a seasoned PM than it is for a junior PM. The PM may use his **experience** with previous projects, information in the company's project **knowledge base** (if there is one…) and **advice** from his peers or from any other person liable to provide help, in particular people (editors, developers, etc.) who have already performed tasks similar to those for which required resources need to be estimated.

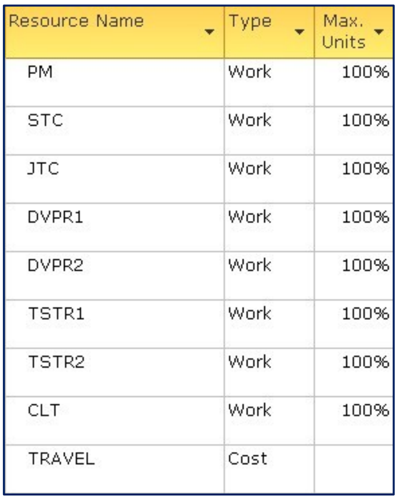
There may be several approaches for the execution of a given task, each with different resource requirements in terms of personnel, equipment, etc. For some tasks, a "**make or buy**" evaluation may be needed to choose between internal and external resources.

The **maximum lead time imposed by the overall schedule** of the project for any given task must be taken into account for the estimation of required resources, since there is a **close relationship between the amount of time required to complete a task and the resources assigned to it**.

Note that there is often a limit to the number of people it is reasonable to assign to any given task: assigning too many resources may be counterproductive due for example to increased communication and coordination complexity.

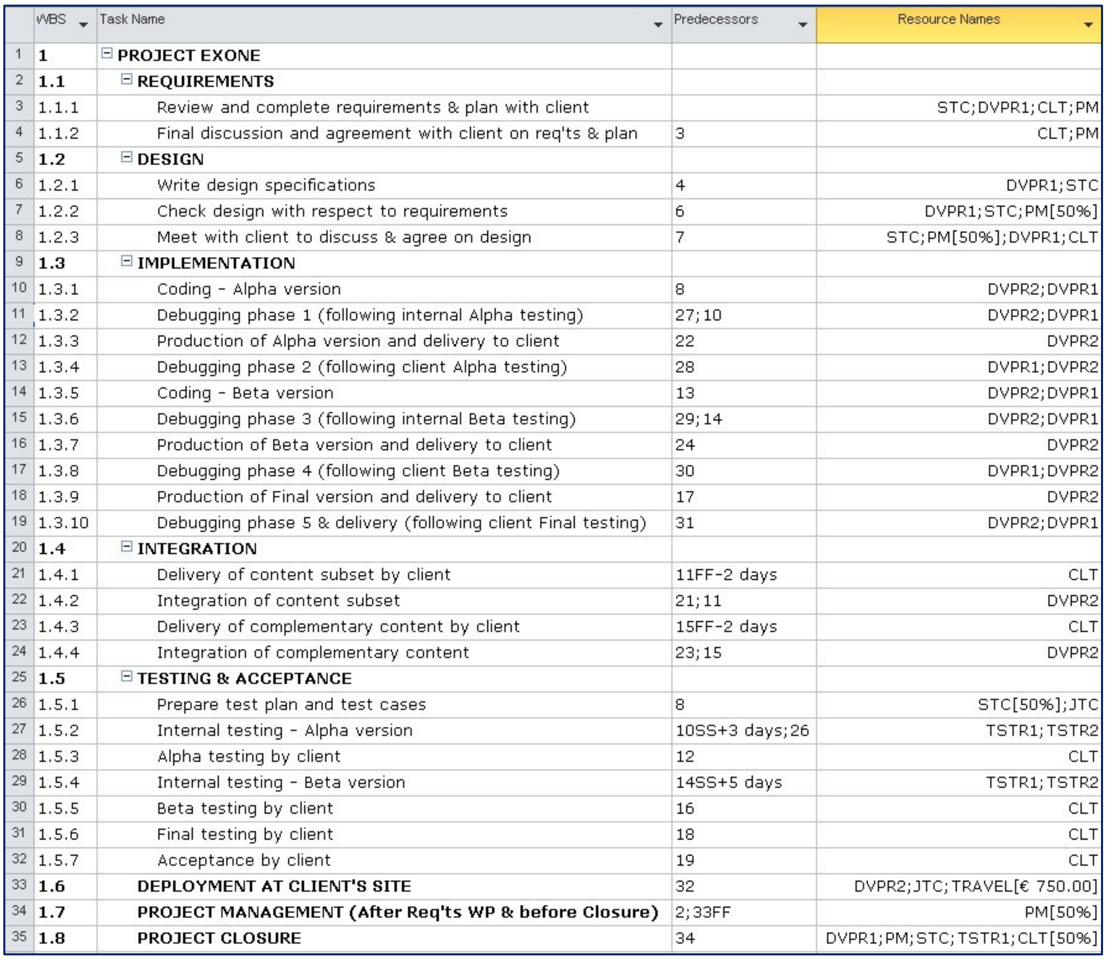
If the possibility of hiring and subcontracting is excluded, the assignment of human resources to tasks will be **constrained** and limited to the "in-house" people available for the project.

In the following Gantt views of the plan for project EXONE, first for the contractor then for the client, the "Resource names" column shows the resources assigned to the various WPs/tasks. The resources are chosen from the list prepared in the “**Resource sheet**”, as shown below for the contractor and on the next page for the client.



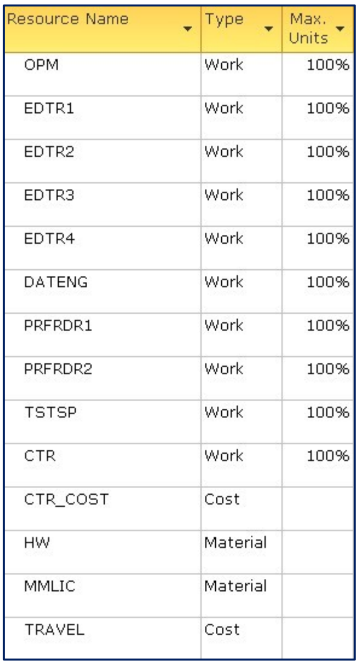
Resource Name	Type	Max. Units
PM	Work	100%
STC	Work	100%
JTC	Work	100%
DVPR1	Work	100%
DVPR2	Work	100%
TSTR1	Work	100%
TSTR2	Work	100%
CLT	Work	100%
TRAVEL	Cost	

Abbreviations are used here, except for TRAVEL, in order to **keep the “Resource Names” column in the Gantt view as narrow as possible** for presentation purposes (PM = project manager; STC = senior technical consultant; JTC = junior technical consultant; DVPR1 & 2 = developers 1 & 2; TSTR1 & 2 = testers 1 & 2; CLT = client).



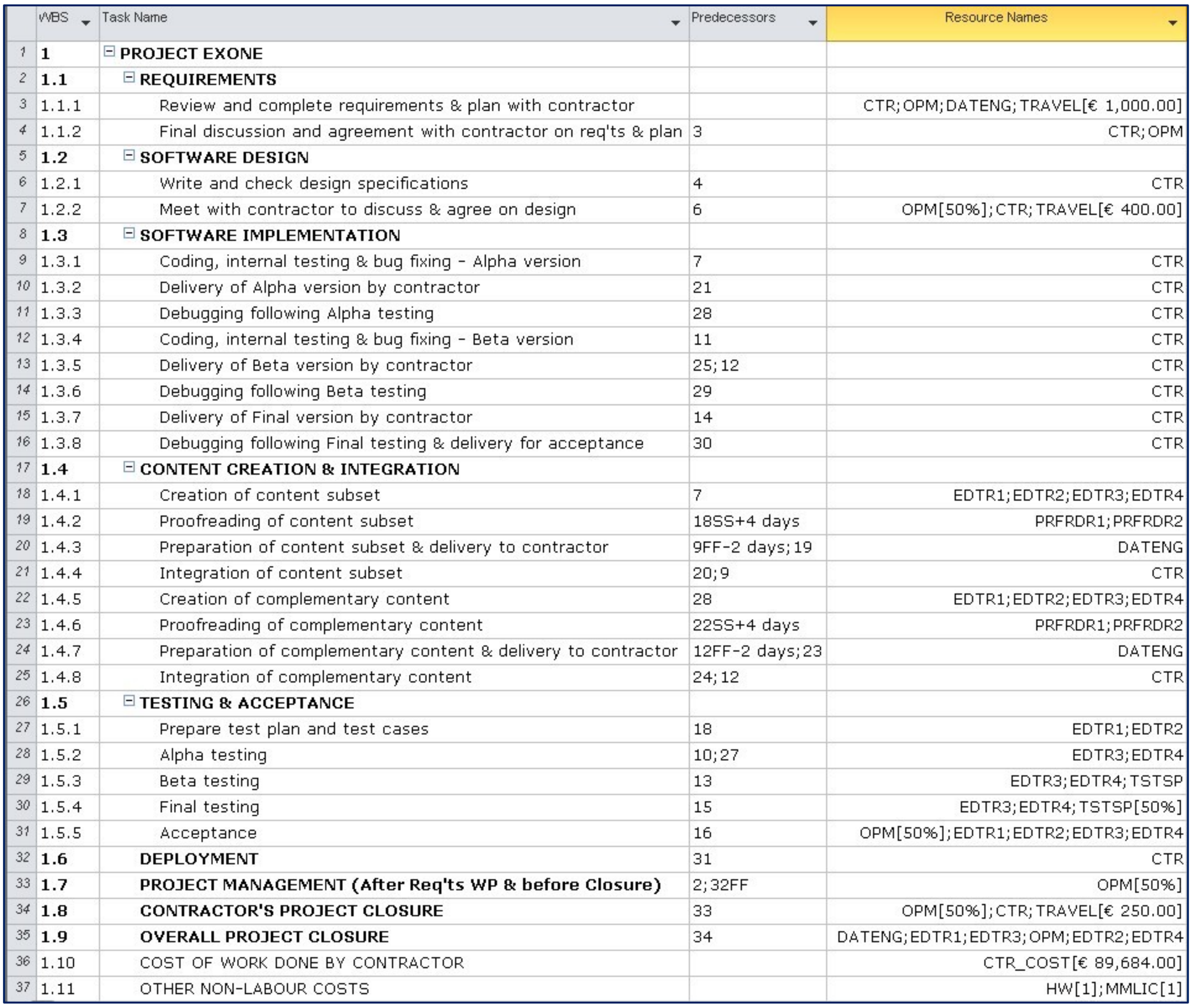
	WBS	Task Name	Predecessors	Resource Names
1	**1**	**PROJECT EXONE**		
2	**1.1**	**REQUIREMENTS**		
3	1.1.1	Review and complete requirements & plan with client		STC;DVPR1;CLT;PM
4	1.1.2	Final discussion and agreement with client on req'ts & plan	3	CLT;PM
5	**1.2**	**DESIGN**		
6	1.2.1	Write design specifications	4	DVPR1;STC
7	1.2.2	Check design with respect to requirements	6	DVPR1;STC;PM[50%]
8	1.2.3	Meet with client to discuss & agree on design	7	STC;PM[50%];DVPR1;CLT
9	**1.3**	**IMPLEMENTATION**		
10	1.3.1	Coding - Alpha version	8	DVPR2;DVPR1
11	1.3.2	Debugging phase 1 (following internal Alpha testing)	27;10	DVPR2;DVPR1
12	1.3.3	Production of Alpha version and delivery to client	22	DVPR2
13	1.3.4	Debugging phase 2 (following client Alpha testing)	28	DVPR1;DVPR2
14	1.3.5	Coding - Beta version	13	DVPR2;DVPR1
15	1.3.6	Debugging phase 3 (following internal Beta testing)	29;14	DVPR2;DVPR1
16	1.3.7	Production of Beta version and delivery to client	24	DVPR2
17	1.3.8	Debugging phase 4 (following client Beta testing)	30	DVPR1;DVPR2
18	1.3.9	Production of Final version and delivery to client	17	DVPR2
19	1.3.10	Debugging phase 5 & delivery (following client Final testing)	31	DVPR2;DVPR1
20	**1.4**	**INTEGRATION**		
21	1.4.1	Delivery of content subset by client	11FF-2 days	CLT
22	1.4.2	Integration of content subset	21;11	DVPR2
23	1.4.3	Delivery of complementary content by client	15FF-2 days	CLT
24	1.4.4	Integration of complementary content	23;15	DVPR2
25	**1.5**	**TESTING & ACCEPTANCE**		
26	1.5.1	Prepare test plan and test cases	8	STC[50%];JTC
27	1.5.2	Internal testing - Alpha version	10SS+3 days;26	TSTR1;TSTR2
28	1.5.3	Alpha testing by client	12	CLT
29	1.5.4	Internal testing - Beta version	14SS+5 days	TSTR1;TSTR2
30	1.5.5	Beta testing by client	16	CLT
31	1.5.6	Final testing by client	18	CLT
32	1.5.7	Acceptance by client	19	CLT
33	**1.6**	**DEPLOYMENT AT CLIENT'S SITE**	32	DVPR2;JTC;TRAVEL[€ 750.00]
34	**1.7**	**PROJECT MANAGEMENT (After Req'ts WP & before Closure)**	2;33FF	PM[50%]
35	**1.8**	**PROJECT CLOSURE**	34	DVPR1;PM;STC;TSTR1;CLT[50%]

Note that the detail of the client's resources does not need to be featured in the contractor's plan, so the use of "CLT" is sufficient. Likewise, in the client's plan, "CTR" is a sufficient representation of the contractor's resources, as featured in the following illustration.



Resource Name	Type	Max. Units
OPM	Work	100%
EDTR1	Work	100%
EDTR2	Work	100%
EDTR3	Work	100%
EDTR4	Work	100%
DATENG	Work	100%
PRFRDR1	Work	100%
PRFRDR2	Work	100%
TSTSP	Work	100%
CTR	Work	100%
CTR_COST	Cost	
HW	Material	
MMLIC	Material	
TRAVEL	Cost	

Like the contractor's resource names, the client's **resource names** are **abbreviated**, with the exception of TRAVEL (OPM = overall project manager; EDTR1, 2, 3 & 4 = editors 1, 2, 3 & 4; DATENG = data engineer; PRFRDR1 & 2 = proofreaders 1 & 2; TSTSP = testing service provider; CTR = contractor, CTR_COST = cost (price) of contractor's work; HW = hardware (& software); MMLIC = multimedia asset licences).



	WBS	Task Name	Predecessors	Resource Names
1	**1**	⊟ **PROJECT EXONE**		
2	**1.1**	⊟ **REQUIREMENTS**		
3	1.1.1	Review and complete requirements & plan with contractor		CTR;OPM;DATENG;TRAVEL[€ 1,000.00]
4	1.1.2	Final discussion and agreement with contractor on req'ts & plan	3	CTR;OPM
5	**1.2**	⊟ **SOFTWARE DESIGN**		
6	1.2.1	Write and check design specifications	4	CTR
7	1.2.2	Meet with contractor to discuss & agree on design	6	OPM[50%];CTR;TRAVEL[€ 400.00]
8	**1.3**	⊟ **SOFTWARE IMPLEMENTATION**		
9	1.3.1	Coding, internal testing & bug fixing - Alpha version	7	CTR
10	1.3.2	Delivery of Alpha version by contractor	21	CTR
11	1.3.3	Debugging following Alpha testing	28	CTR
12	1.3.4	Coding, internal testing & bug fixing - Beta version	11	CTR
13	1.3.5	Delivery of Beta version by contractor	25;12	CTR
14	1.3.6	Debugging following Beta testing	29	CTR
15	1.3.7	Delivery of Final version by contractor	14	CTR
16	1.3.8	Debugging following Final testing & delivery for acceptance	30	CTR
17	**1.4**	⊟ **CONTENT CREATION & INTEGRATION**		
18	1.4.1	Creation of content subset	7	EDTR1;EDTR2;EDTR3;EDTR4
19	1.4.2	Proofreading of content subset	18SS+4 days	PRFRDR1;PRFRDR2
20	1.4.3	Preparation of content subset & delivery to contractor	9FF-2 days;19	DATENG
21	1.4.4	Integration of content subset	20;9	CTR
22	1.4.5	Creation of complementary content	28	EDTR1;EDTR2;EDTR3;EDTR4
23	1.4.6	Proofreading of complementary content	22SS+4 days	PRFRDR1;PRFRDR2
24	1.4.7	Preparation of complementary content & delivery to contractor	12FF-2 days;23	DATENG
25	1.4.8	Integration of complementary content	24;12	CTR
26	**1.5**	⊟ **TESTING & ACCEPTANCE**		
27	1.5.1	Prepare test plan and test cases	18	EDTR1;EDTR2
28	1.5.2	Alpha testing	10;27	EDTR3;EDTR4
29	1.5.3	Beta testing	13	EDTR3;EDTR4;TSTSP
30	1.5.4	Final testing	15	EDTR3;EDTR4;TSTSP[50%]
31	1.5.5	Acceptance	16	OPM[50%];EDTR1;EDTR2;EDTR3;EDTR4
32	**1.6**	**DEPLOYMENT**	31	CTR
33	**1.7**	**PROJECT MANAGEMENT (After Req'ts WP & before Closure)**	2;32FF	OPM[50%]
34	**1.8**	**CONTRACTOR'S PROJECT CLOSURE**	33	OPM[50%];CTR;TRAVEL[€ 250.00]
35	**1.9**	**OVERALL PROJECT CLOSURE**	34	DATENG;EDTR1;EDTR3;OPM;EDTR2;EDTR4
36	1.10	COST OF WORK DONE BY CONTRACTOR		CTR_COST[€ 89,684.00]
37	1.11	OTHER NON-LABOUR COSTS		HW[1];MMLIC[1]

Like task predecessors, **resources are always assigned at the lowest level of the hierarchical list**, ie at task level, or WP level if the WP has no explicit subordinate tasks.

Note that **resources may not or need not be available 100% of their time**.
For example, in the contractor's plan for project EXONE the STC is expected to spend only 50% of his time (coded as "STC[50%]") on test planning and writing test cases, and the PM is assumed to dedicate 100% of his time to the project during its Requirements and Closure phases, but only 50% of his time (coded as "PM[50%]") during the other phases.

Also note that, in principle, **people should not work more than 100% of their time!** For example, in the client's plan for project EXONE "OPM[50%]" has been assigned to tasks 1.2.2 and 1.5.5 in order to avoid overloading the OPM, who is also busy 50% of his time with WP 1.7, namely "(OVERALL) PROJECT MANAGEMENT".

Resource overload is a frequent error made in project planning, which the PM should be careful to avoid (apart from justified exceptions…). The "**Resource usage**" function of MS Project or ProjectLibre can be very helpful in identifying such overload situations.

A **simulation**, in "real-life" conditions or on paper, may be needed in order to estimate resource requirements for certain tasks. Project management software applications can be used to make such simulations, but a spreadsheet is often quite sufficient.

> *One of the editorial WPs of the EHM project included a task which consisted in indexing each encyclopedic article by assigning one or several "triplets" of metadata to it, each triplet including a topic, a geographical location and a period. Because there was no previous experience of such a task, a group of editors performed it on a representative sample of articles, using editing tools developed for that purpose. The average amount of time required to index an article was measured. As a result, the number of persons required for the task was easily calculated, given the maximum duration of 6 months imposed by the overall schedule of the project.*
>
> - *Number of articles to index: 50,000*
> - *Number of minutes per article (in the sample): 5*
> - *Total number of minutes required: 250,000*
> - *Margin of error (as a precaution…): 15%*
> - *Number of minutes required after adjustment: 287,500*
> - *Number of hours: 4,792*
> - *Number of 7-hour days: 685*
> - *Number of 5-day weeks: 137*
> - *Number of 4-week months: 34 (A)*
> - *Maximum lead time in months: 6 (B)*
> - *Number of persons required: 6 (= A/B, rounded)*
>
> *This estimate led to the hiring of 5 "editors-indexers" with a 6-month temporary contract (the 6th person required was already on board).*
> *The progress of this activity was measured on a regular and frequent basis, in order to detect any possible deviation with respect to the schedule that had been fixed.*

Apart from **human resources** (in-house as well as external), **other types of resources** are generally required for the execution of a project, for example (non-exhaustive list):

- offices (building, floor space, furniture, utilities, etc.),
- workstations (computers and software licences),
- printers, scanners, audiovisual equipment,
- servers, hosting service, domain name, etc.,
- consumables,
- documentation (reference books, etc.),
- multimedia asset licences
- travel & accommodation (eg for certain meetings).

Contractors may have equipment of their own. In some cases however, the client will need to provide equipment which is specifically required for the subcontracted work.

> *Two servers required for the EHM "back office" were purchased by Hachette and installed at the contractor's facility for the duration of the development and debugging of the tools. They were then moved and put into operation at Hachette.*

The result of the "Estimate task resources" process is a set of additional attributes for tasks in the task list and a "**resource schedule**" providing the list of all resources required for the project. The list is generally organized by type of resource: people (with a definition of their skills, roles and responsibilities), equipment, materials, etc.

The "**Resource sheet**" of a plan built with MS Project or ProjectLibre identifies human resources by their names and/or abbreviations and other attributes (eg percentage of time spent on the project, cost), as shown further on in this chapter.

As shown on previous pages in the examples for project EXONE, resources other than people, which may be called "**non-labour resources**", can also be featured in the resource sheet of a project management application, and will need to be assigned to existing tasks, or to "dummy" tasks specially created for cost calculation purposes.
For example, in the contractor's plan for project EXONE, "TRAVEL" has been assigned to the "DEPLOYMENT AT CLIENT'S SITE" WP since DVPR2 and STC will need to travel to the client's site to do the corresponding work.
In the client's plan for project EXONE, "TRAVEL" has been assigned to those tasks/WPs corresponding to meetings for which OPM and DATENG need to travel to the contractor's place of work. The "CTR_COST" resource, namely the cost (price) of the work to be done by the contractor, and the "HW" and "MMLIC" resources have been assigned to dummy tasks/WPs (1.10 & 1.11) so that their costs are taken into account by the project management software application for the calculation of the total cost of the project, as illustrated in the "Estimate costs" section of this chapter.

Estimate task durations

This process consists in estimating the **duration** (ie **lead time) required to complete the work to be done** for each task, **with the resources assigned to it**.

"Estimate task durations" and "Estimate task resources" processes are of course closely related. Several **iterations** may be required in order to complete these processes.
For some work packages or tasks, **durations may be imposed** and therefore need to be accepted as **planning constraints**.
For example, the project owner may require a prototype to be available within 6 weeks from the start of project execution, regardless of the number and availability of resources for the corresponding work. As another example of scheduling constraint, resources required for a given task may be available to work on the task only within certain time frames.

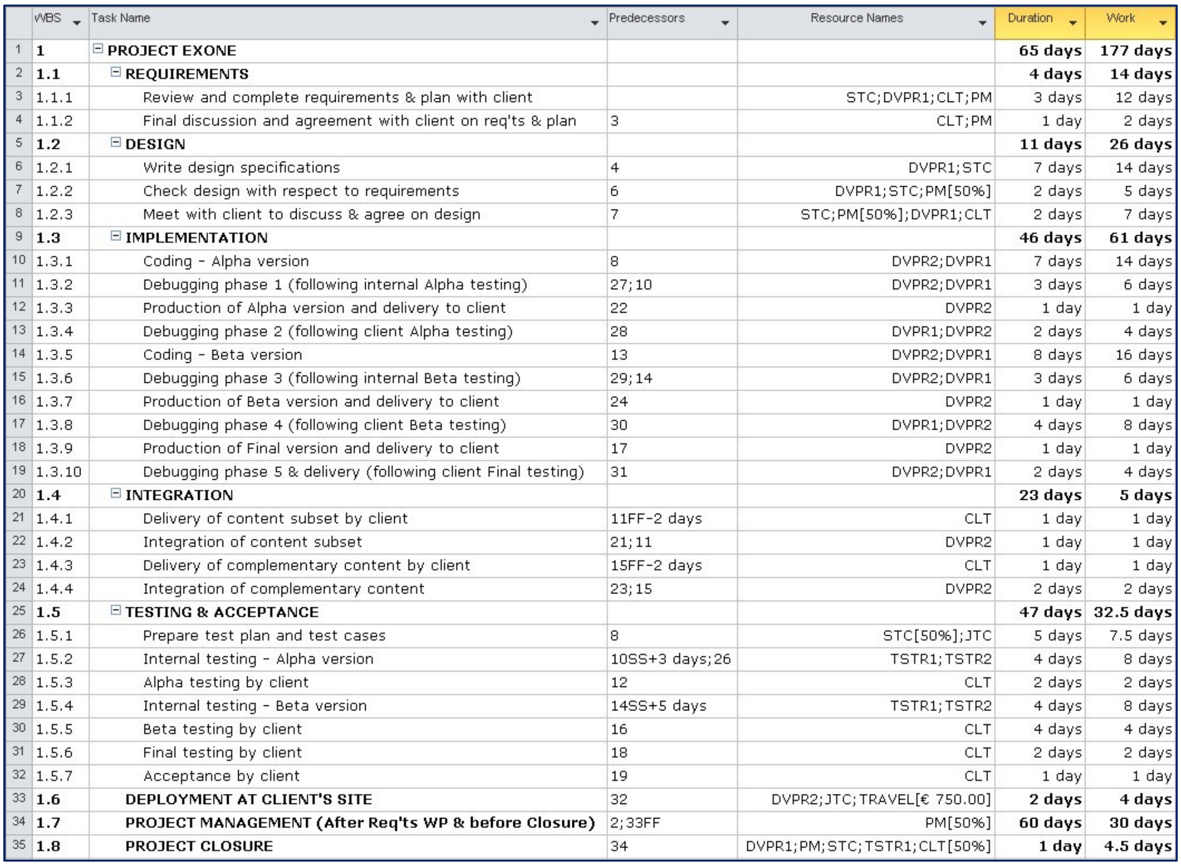
Estimating the **workload** (number of persons x duration) for a given task may sometimes be easier than estimating its duration. In this case, the task duration will be derived from the workload and the resources assigned to the task.

As mentioned previously, human resources may not or need not be available 100% of their time for a given task, which obviously has an impact on the duration of the task.
For example, if a task for which a workload of 10 "person-days" has been estimated is assigned to a person working only the first 4 days of any week, then the actual lead time for the completion of this task will be 12 working days. The **duration** attributed to this task should therefore be 12 days, whereas the actual **workload** is 10 person-days.

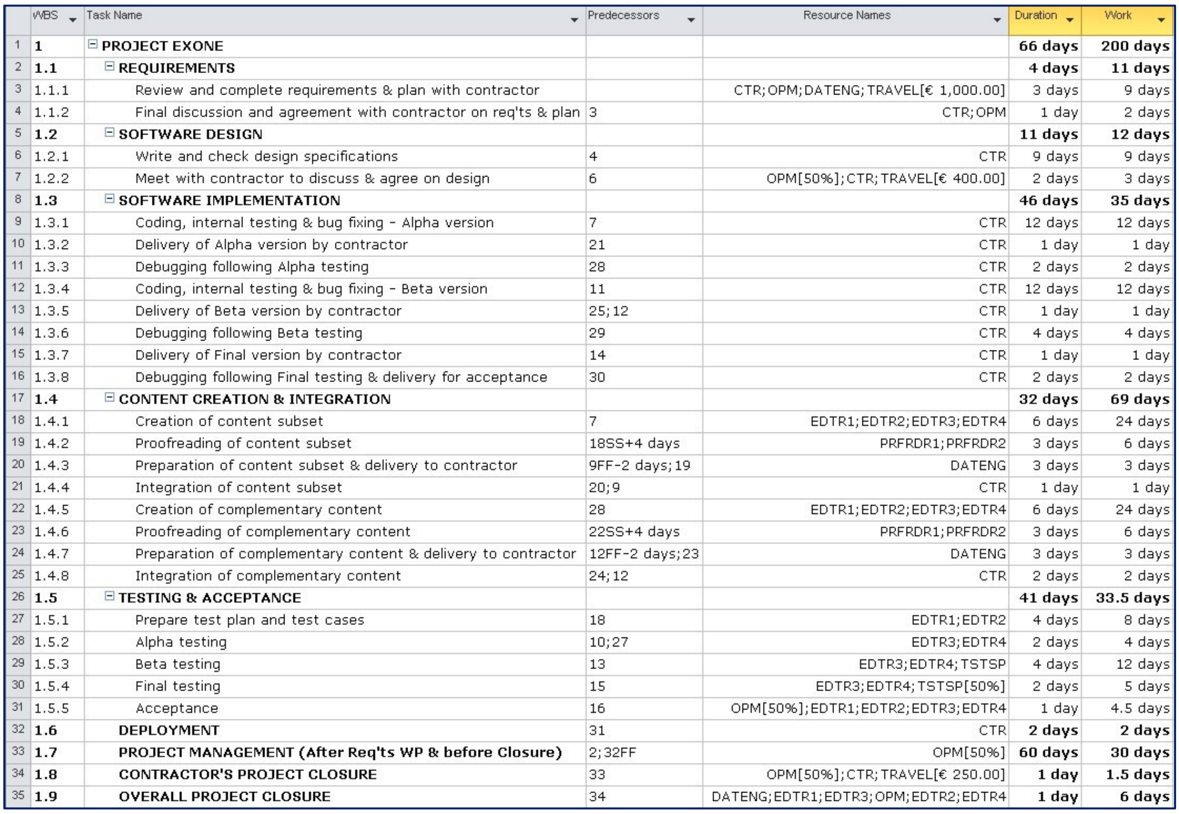
It is the **workload**, not the duration, which is taken into account for **cost calculation**.

Note that **durations** are always expressed in units of **work time** (workdays, workweeks, etc.), ie **excluding weekends and holidays** (except of course if the project requires working on some or all weekends and/or holidays…).

Project management applications automatically calculate workloads from duration and resource information (or calculate durations from workload and resource information), as shown in the following examples for project EXONE (contractor's view then client's view).



	WBS	Task Name	Predecessors	Resource Names	Duration	Work
1	**1**	⊟ **PROJECT EXONE**			**65 days**	**177 days**
2	**1.1**	⊟ **REQUIREMENTS**			**4 days**	**14 days**
3	1.1.1	Review and complete requirements & plan with client		STC;DVPR1;CLT;PM	3 days	12 days
4	1.1.2	Final discussion and agreement with client on req'ts & plan	3	CLT;PM	1 day	2 days
5	**1.2**	⊟ **DESIGN**			**11 days**	**26 days**
6	1.2.1	Write design specifications	4	DVPR1;STC	7 days	14 days
7	1.2.2	Check design with respect to requirements	6	DVPR1;STC;PM[50%]	2 days	5 days
8	1.2.3	Meet with client to discuss & agree on design	7	STC;PM[50%];DVPR1;CLT	2 days	7 days
9	**1.3**	⊟ **IMPLEMENTATION**			**46 days**	**61 days**
10	1.3.1	Coding - Alpha version	8	DVPR2;DVPR1	7 days	14 days
11	1.3.2	Debugging phase 1 (following internal Alpha testing)	27;10	DVPR2;DVPR1	3 days	6 days
12	1.3.3	Production of Alpha version and delivery to client	22	DVPR2	1 day	1 day
13	1.3.4	Debugging phase 2 (following client Alpha testing)	28	DVPR1;DVPR2	2 days	4 days
14	1.3.5	Coding - Beta version	13	DVPR2;DVPR1	8 days	16 days
15	1.3.6	Debugging phase 3 (following internal Beta testing)	29;14	DVPR2;DVPR1	3 days	6 days
16	1.3.7	Production of Beta version and delivery to client	24	DVPR2	1 day	1 day
17	1.3.8	Debugging phase 4 (following client Beta testing)	30	DVPR1;DVPR2	4 days	8 days
18	1.3.9	Production of Final version and delivery to client	17	DVPR2	1 day	1 day
19	1.3.10	Debugging phase 5 & delivery (following client Final testing)	31	DVPR2;DVPR1	2 days	4 days
20	**1.4**	⊟ **INTEGRATION**			**23 days**	**5 days**
21	1.4.1	Delivery of content subset by client	11FF-2 days	CLT	1 day	1 day
22	1.4.2	Integration of content subset	21;11	DVPR2	1 day	1 day
23	1.4.3	Delivery of complementary content by client	15FF-2 days	CLT	1 day	1 day
24	1.4.4	Integration of complementary content	23;15	DVPR2	2 days	2 days
25	**1.5**	⊟ **TESTING & ACCEPTANCE**			**47 days**	**32.5 days**
26	1.5.1	Prepare test plan and test cases	8	STC[50%];JTC	5 days	7.5 days
27	1.5.2	Internal testing - Alpha version	10SS+3 days;26	TSTR1;TSTR2	4 days	8 days
28	1.5.3	Alpha testing by client	12	CLT	2 days	2 days
29	1.5.4	Internal testing - Beta version	14SS+5 days	TSTR1;TSTR2	4 days	8 days
30	1.5.5	Beta testing by client	16	CLT	4 days	4 days
31	1.5.6	Final testing by client	18	CLT	2 days	2 days
32	1.5.7	Acceptance by client	19	CLT	1 day	1 day
33	**1.6**	**DEPLOYMENT AT CLIENT'S SITE**	32	DVPR2;JTC;TRAVEL[€ 750.00]	**2 days**	**4 days**
34	**1.7**	**PROJECT MANAGEMENT (After Req'ts WP & before Closure)**	2;33FF	PM[50%]	**60 days**	**30 days**
35	**1.8**	**PROJECT CLOSURE**	34	DVPR1;PM;STC;TSTR1;CLT[50%]	**1 day**	**4.5 days**



	WBS	Task Name	Predecessors	Resource Names	Duration	Work
1	**1**	⊟ **PROJECT EXONE**			**66 days**	**200 days**
2	**1.1**	⊟ **REQUIREMENTS**			**4 days**	**11 days**
3	1.1.1	Review and complete requirements & plan with contractor		CTR;OPM;DATENG;TRAVEL[€ 1,000.00]	3 days	9 days
4	1.1.2	Final discussion and agreement with contractor on req'ts & plan	3	CTR;OPM	1 day	2 days
5	**1.2**	⊟ **SOFTWARE DESIGN**			**11 days**	**12 days**
6	1.2.1	Write and check design specifications	4	CTR	9 days	9 days
7	1.2.2	Meet with contractor to discuss & agree on design	6	OPM[50%];CTR;TRAVEL[€ 400.00]	2 days	3 days
8	**1.3**	⊟ **SOFTWARE IMPLEMENTATION**			**46 days**	**35 days**
9	1.3.1	Coding, internal testing & bug fixing - Alpha version	7	CTR	12 days	12 days
10	1.3.2	Delivery of Alpha version by contractor	21	CTR	1 day	1 day
11	1.3.3	Debugging following Alpha testing	28	CTR	2 days	2 days
12	1.3.4	Coding, internal testing & bug fixing - Beta version	11	CTR	12 days	12 days
13	1.3.5	Delivery of Beta version by contractor	25;12	CTR	1 day	1 day
14	1.3.6	Debugging following Beta testing	29	CTR	4 days	4 days
15	1.3.7	Delivery of Final version by contractor	14	CTR	1 day	1 day
16	1.3.8	Debugging following Final testing & delivery for acceptance	30	CTR	2 days	2 days
17	**1.4**	⊟ **CONTENT CREATION & INTEGRATION**			**32 days**	**69 days**
18	1.4.1	Creation of content subset	7	EDTR1;EDTR2;EDTR3;EDTR4	6 days	24 days
19	1.4.2	Proofreading of content subset	18SS+4 days	PRFRDR1;PRFRDR2	3 days	6 days
20	1.4.3	Preparation of content subset & delivery to contractor	9FF-2 days;19	DATENG	3 days	3 days
21	1.4.4	Integration of content subset	20;9	CTR	1 day	1 day
22	1.4.5	Creation of complementary content	28	EDTR1;EDTR2;EDTR3;EDTR4	6 days	24 days
23	1.4.6	Proofreading of complementary content	22SS+4 days	PRFRDR1;PRFRDR2	3 days	6 days
24	1.4.7	Preparation of complementary content & delivery to contractor	12FF-2 days;23	DATENG	3 days	3 days
25	1.4.8	Integration of complementary content	24;12	CTR	2 days	2 days
26	**1.5**	⊟ **TESTING & ACCEPTANCE**			**41 days**	**33.5 days**
27	1.5.1	Prepare test plan and test cases	18	EDTR1;EDTR2	4 days	8 days
28	1.5.2	Alpha testing	10;27	EDTR3;EDTR4	2 days	4 days
29	1.5.3	Beta testing	13	EDTR3;EDTR4;TSTSP	4 days	12 days
30	1.5.4	Final testing	15	EDTR3;EDTR4;TSTSP[50%]	2 days	5 days
31	1.5.5	Acceptance	16	OPM[50%];EDTR1;EDTR2;EDTR3;EDTR4	1 day	4.5 days
32	**1.6**	**DEPLOYMENT**	31	CTR	**2 days**	**2 days**
33	**1.7**	**PROJECT MANAGEMENT (After Req'ts WP & before Closure)**	2;32FF	OPM[50%]	**60 days**	**30 days**
34	**1.8**	**CONTRACTOR'S PROJECT CLOSURE**	33	OPM[50%];CTR;TRAVEL[€ 250.00]	**1 day**	**1.5 days**
35	**1.9**	**OVERALL PROJECT CLOSURE**	34	DATENG;EDTR1;EDTR3;OPM;EDTR2;EDTR4	**1 day**	**6 days**

Note that, like task predecessors and resources, **durations are always assigned at the lowest level of the hierarchical list**, ie at task level, or WP level if the WP has no explicit subordinate tasks.

Also note that **the duration of a work package (WP) is not always the sum of the durations of its subordinate tasks**, since such tasks are not necessarily strictly consecutive or contiguous, ie there may be **parallelism or gaps** between them.
For example, with project EXONE the "(SOFTWARE) IMPLEMENTATION" WP has a total duration of 46 workdays, not 32 workdays, which is the sum of the durations of all tasks in that particular WP.

The "Work" column in the previous tables shows the **workload** (in person-days) calculated for each **task**, for each **work package** and for the **project** as a whole.
For example, WP 1.8 in the contractor's plan has a workload of 4.5 person-days because there are 4 people assigned to it full-time and one resource ("CLT") part-time (50%) for a duration of 1 workday.

Duration estimates need to take into account **time** that may be required **for learning**, since people usually need to familiarize themselves with new tools, techniques and procedures.

Among **tools and techniques for estimating task durations or workloads**, the following are widely used and generally combined:

- expert judgment,
- analogous estimating,
- parametric estimating,
- three-point estimates,
- reserve analysis.

Expert judgment is based on experience of previous projects and information from people (editors, developers, etc.) who have already performed tasks similar to those for which an estimate needs to be made.

Analogous estimating uses information on the duration or workload (and other parameters) of similar tasks in previous projects.

Parametric estimating is an extrapolation based on statistical data that establishes the amount of time needed for a "unit of work". Input data may be historical data or the result of a simulation (eg as described previously for the indexing task of the EHM project).

Three-point estimates: this technique, which is worth what it is worth (no less, no more!), consists in calculating the expected duration (Te) or workload of a task as the weighted average of the most likely duration (Tm) or workload estimate, which is given a weight of 4, and of the optimistic and pessimistic estimates (To and Tp), which are each given a weight of 1. The corresponding formula is:

- Te = (To + 4 x Tm + Tp) / 6

Reserve analysis relates to the natural uncertainty of task duration or workload estimates. Uncertainty may lead to the inclusion of "**contingency reserves**" (also called "**time reserves**" or "**buffers**") in the estimates. For example, a percentage of the estimated duration or a number of work periods (eg days or weeks) may be added to the estimate as a contingency reserve.

The result of the "Estimate task durations" process is a set of additional attributes for the tasks in the task list.

Develop the schedule

The project schedule is derived from the **sequence of tasks**, their **durations** and **resources**, as well as **constraints** such as **milestones** (start or finish dates) that have been **fixed** (eg imposed by the project owner) for certain tasks or work packages.
A typical example of constraint is the **product launch date**, which may be used as a "starting point", so to speak, for "**backward planning**". Another example is that of data delivery which cannot take place before a certain date imposed by the content provider.

The **initial schedule** serves as a **baseline** to track project progress. The schedule generally needs to be revised as the project moves forward, but the initial schedule should be kept as a **reference**.

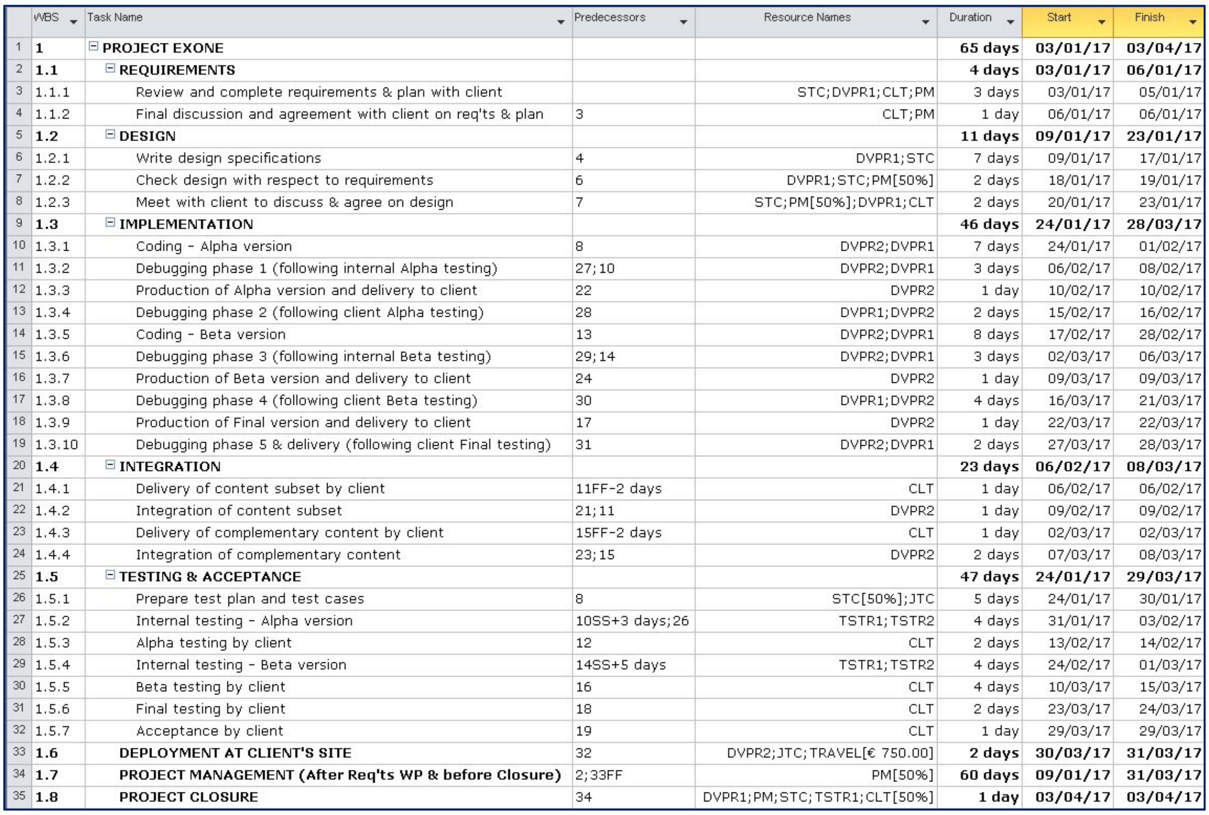
Major schedule revisions need to be approved by all stakeholders concerned.
One important notion in a project schedule is that of "**critical path**", ie sequences of tasks that determine the project's finish date. For critical-path tasks, there is no "**float**", ie no **margin of error**. Any delay in completing these tasks has a **direct impact** on the project's finish date.

Project management applications, such as MS Project and ProjectLibre, are very helpful for creating a project's schedule and for displaying critical paths.

Less specialized tools, such as a **spreadsheet**, may however be used for relatively simple project schedules or to provide a high-level view of the schedule of more complex projects, as illustrated further on.

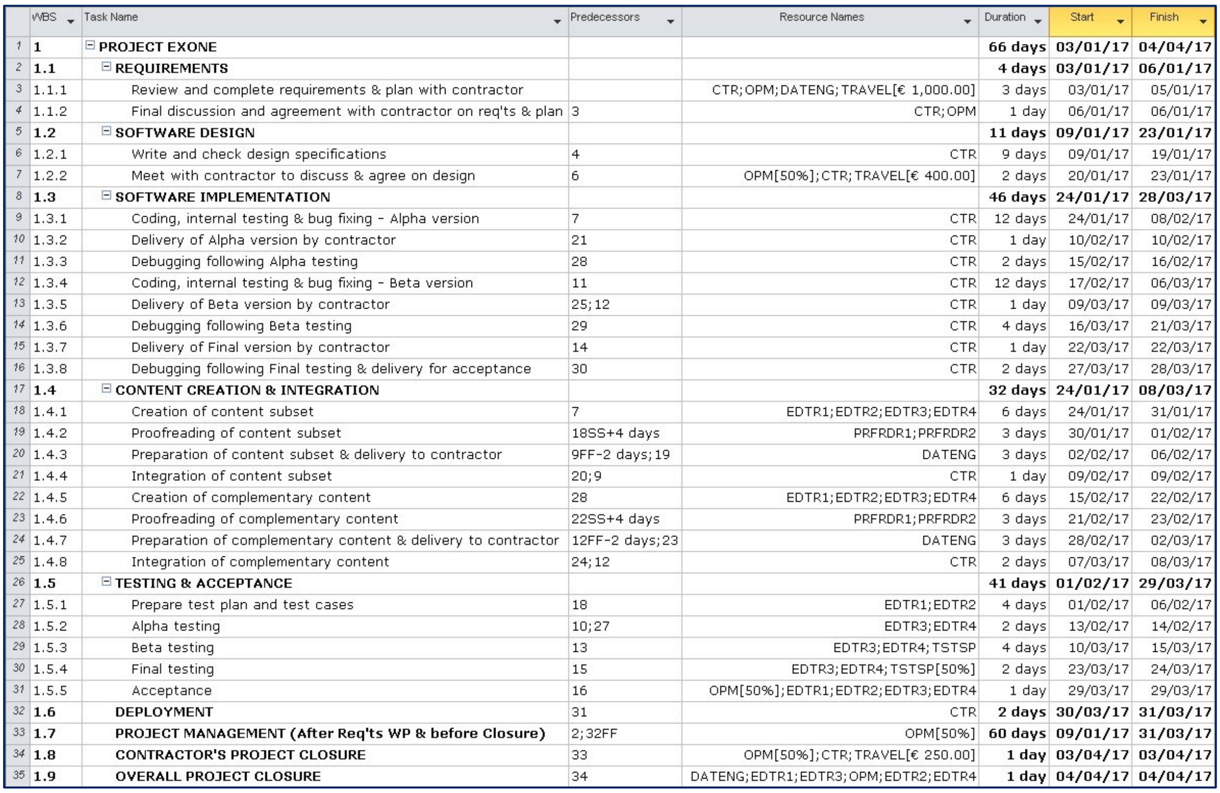
The following two illustrations show the Gantt tabular views for project EXONE, including "Start" and "Finish" dates.

- Project EXONE – Contractor's view:



	WBS	Task Name	Predecessors	Resource Names	Duration	Start	Finish
1	**1**	⊟ **PROJECT EXONE**			**65 days**	**03/01/17**	**03/04/17**
2	**1.1**	⊟ **REQUIREMENTS**			**4 days**	**03/01/17**	**06/01/17**
3	1.1.1	Review and complete requirements & plan with client		STC;DVPR1;CLT;PM	3 days	03/01/17	05/01/17
4	1.1.2	Final discussion and agreement with client on req'ts & plan	3	CLT;PM	1 day	06/01/17	06/01/17
5	**1.2**	⊟ **DESIGN**			**11 days**	**09/01/17**	**23/01/17**
6	1.2.1	Write design specifications	4	DVPR1;STC	7 days	09/01/17	17/01/17
7	1.2.2	Check design with respect to requirements	6	DVPR1;STC;PM[50%]	2 days	18/01/17	19/01/17
8	1.2.3	Meet with client to discuss & agree on design	7	STC;PM[50%];DVPR1;CLT	2 days	20/01/17	23/01/17
9	**1.3**	⊟ **IMPLEMENTATION**			**46 days**	**24/01/17**	**28/03/17**
10	1.3.1	Coding - Alpha version	8	DVPR2;DVPR1	7 days	24/01/17	01/02/17
11	1.3.2	Debugging phase 1 (following internal Alpha testing)	27;10	DVPR2;DVPR1	3 days	06/02/17	08/02/17
12	1.3.3	Production of Alpha version and delivery to client	22	DVPR2	1 day	10/02/17	10/02/17
13	1.3.4	Debugging phase 2 (following client Alpha testing)	28	DVPR1;DVPR2	2 days	15/02/17	16/02/17
14	1.3.5	Coding - Beta version	13	DVPR2;DVPR1	8 days	17/02/17	28/02/17
15	1.3.6	Debugging phase 3 (following internal Beta testing)	29;14	DVPR2;DVPR1	3 days	02/03/17	06/03/17
16	1.3.7	Production of Beta version and delivery to client	24	DVPR2	1 day	09/03/17	09/03/17
17	1.3.8	Debugging phase 4 (following client Beta testing)	30	DVPR1;DVPR2	4 days	16/03/17	21/03/17
18	1.3.9	Production of Final version and delivery to client	17	DVPR2	1 day	22/03/17	22/03/17
19	1.3.10	Debugging phase 5 & delivery (following client Final testing)	31	DVPR2;DVPR1	2 days	27/03/17	28/03/17
20	**1.4**	⊟ **INTEGRATION**			**23 days**	**06/02/17**	**08/03/17**
21	1.4.1	Delivery of content subset by client	11FF-2 days	CLT	1 day	06/02/17	06/02/17
22	1.4.2	Integration of content subset	21;11	DVPR2	1 day	09/02/17	09/02/17
23	1.4.3	Delivery of complementary content by client	15FF-2 days	CLT	1 day	02/03/17	02/03/17
24	1.4.4	Integration of complementary content	23;15	DVPR2	2 days	07/03/17	08/03/17
25	**1.5**	⊟ **TESTING & ACCEPTANCE**			**47 days**	**24/01/17**	**29/03/17**
26	1.5.1	Prepare test plan and test cases	8	STC[50%];JTC	5 days	24/01/17	30/01/17
27	1.5.2	Internal testing - Alpha version	10SS+3 days;26	TSTR1;TSTR2	4 days	31/01/17	03/02/17
28	1.5.3	Alpha testing by client	12	CLT	2 days	13/02/17	14/02/17
29	1.5.4	Internal testing - Beta version	14SS+5 days	TSTR1;TSTR2	4 days	24/02/17	01/03/17
30	1.5.5	Beta testing by client	16	CLT	4 days	10/03/17	15/03/17
31	1.5.6	Final testing by client	18	CLT	2 days	23/03/17	24/03/17
32	1.5.7	Acceptance by client	19	CLT	1 day	29/03/17	29/03/17
33	**1.6**	**DEPLOYMENT AT CLIENT'S SITE**	32	DVPR2;JTC;TRAVEL[€ 750.00]	**2 days**	**30/03/17**	**31/03/17**
34	**1.7**	**PROJECT MANAGEMENT (After Req'ts WP & before Closure)**	2;33FF	PM[50%]	**60 days**	**09/01/17**	**31/03/17**
35	**1.8**	**PROJECT CLOSURE**	34	DVPR1;PM;STC;TSTR1;CLT[50%]	**1 day**	**03/04/17**	**03/04/17**

- Project EXONE – Client's view:

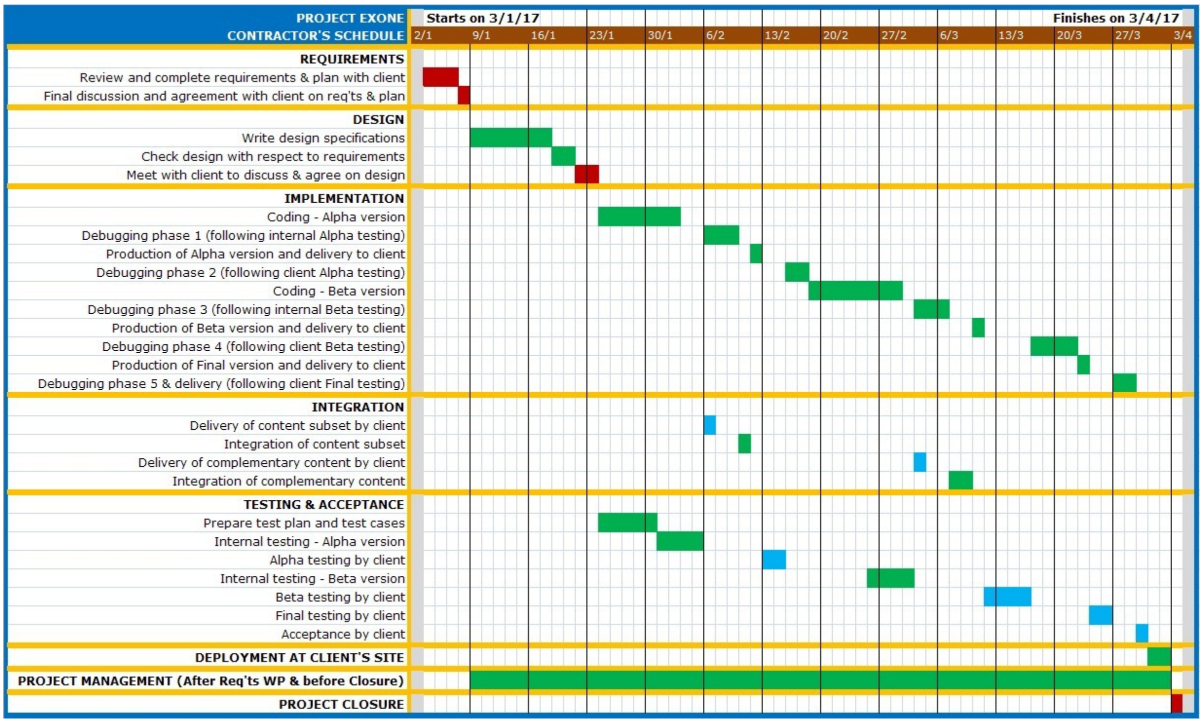


	WBS	Task Name	Predecessors	Resource Names	Duration	Start	Finish
1	**1**	⊟ **PROJECT EXONE**			**66 days**	**03/01/17**	**04/04/17**
2	**1.1**	⊟ **REQUIREMENTS**			**4 days**	**03/01/17**	**06/01/17**
3	1.1.1	Review and complete requirements & plan with contractor		CTR;OPM;DATENG;TRAVEL[€ 1,000.00]	3 days	03/01/17	05/01/17
4	1.1.2	Final discussion and agreement with contractor on req'ts & plan	3	CTR;OPM	1 day	06/01/17	06/01/17
5	**1.2**	⊟ **SOFTWARE DESIGN**			**11 days**	**09/01/17**	**23/01/17**
6	1.2.1	Write and check design specifications	4	CTR	9 days	09/01/17	19/01/17
7	1.2.2	Meet with contractor to discuss & agree on design	6	OPM[50%];CTR;TRAVEL[€ 400.00]	2 days	20/01/17	23/01/17
8	**1.3**	⊟ **SOFTWARE IMPLEMENTATION**			**46 days**	**24/01/17**	**28/03/17**
9	1.3.1	Coding, internal testing & bug fixing - Alpha version	7	CTR	12 days	24/01/17	08/02/17
10	1.3.2	Delivery of Alpha version by contractor	21	CTR	1 day	10/02/17	10/02/17
11	1.3.3	Debugging following Alpha testing	28	CTR	2 days	15/02/17	16/02/17
12	1.3.4	Coding, internal testing & bug fixing - Beta version	11	CTR	12 days	17/02/17	06/03/17
13	1.3.5	Delivery of Beta version by contractor	25;12	CTR	1 day	09/03/17	09/03/17
14	1.3.6	Debugging following Beta testing	29	CTR	4 days	16/03/17	21/03/17
15	1.3.7	Delivery of Final version by contractor	14	CTR	1 day	22/03/17	22/03/17
16	1.3.8	Debugging following Final testing & delivery for acceptance	30	CTR	2 days	27/03/17	28/03/17
17	**1.4**	⊟ **CONTENT CREATION & INTEGRATION**			**32 days**	**24/01/17**	**08/03/17**
18	1.4.1	Creation of content subset	7	EDTR1;EDTR2;EDTR3;EDTR4	6 days	24/01/17	31/01/17
19	1.4.2	Proofreading of content subset	18SS+4 days	PRFRDR1;PRFRDR2	3 days	30/01/17	01/02/17
20	1.4.3	Preparation of content subset & delivery to contractor	9FF-2 days;19	DATENG	3 days	02/02/17	06/02/17
21	1.4.4	Integration of content subset	20;9	CTR	1 day	09/02/17	09/02/17
22	1.4.5	Creation of complementary content	28	EDTR1;EDTR2;EDTR3;EDTR4	6 days	15/02/17	22/02/17
23	1.4.6	Proofreading of complementary content	22SS+4 days	PRFRDR1;PRFRDR2	3 days	21/02/17	23/02/17
24	1.4.7	Preparation of complementary content & delivery to contractor	12FF-2 days;23	DATENG	3 days	28/02/17	02/03/17
25	1.4.8	Integration of complementary content	24;12	CTR	2 days	07/03/17	08/03/17
26	**1.5**	⊟ **TESTING & ACCEPTANCE**			**41 days**	**01/02/17**	**29/03/17**
27	1.5.1	Prepare test plan and test cases	18	EDTR1;EDTR2	4 days	01/02/17	06/02/17
28	1.5.2	Alpha testing	10;27	EDTR3;EDTR4	2 days	13/02/17	14/02/17
29	1.5.3	Beta testing	13	EDTR3;EDTR4;TSTSP	4 days	10/03/17	15/03/17
30	1.5.4	Final testing	15	EDTR3;EDTR4;TSTSP[50%]	2 days	23/03/17	24/03/17
31	1.5.5	Acceptance	16	OPM[50%];EDTR1;EDTR2;EDTR3;EDTR4	1 day	29/03/17	29/03/17
32	**1.6**	**DEPLOYMENT**	31	CTR	**2 days**	**30/03/17**	**31/03/17**
33	**1.7**	**PROJECT MANAGEMENT (After Req'ts WP & before Closure)**	2;32FF	OPM[50%]	**60 days**	**09/01/17**	**31/03/17**
34	**1.8**	**CONTRACTOR'S PROJECT CLOSURE**	33	OPM[50%];CTR;TRAVEL[€ 250.00]	**1 day**	**03/04/17**	**03/04/17**
35	**1.9**	**OVERALL PROJECT CLOSURE**	34	DATENG;EDTR1;EDTR3;OPM;EDTR2;EDTR4	**1 day**	**04/04/17**	**04/04/17**

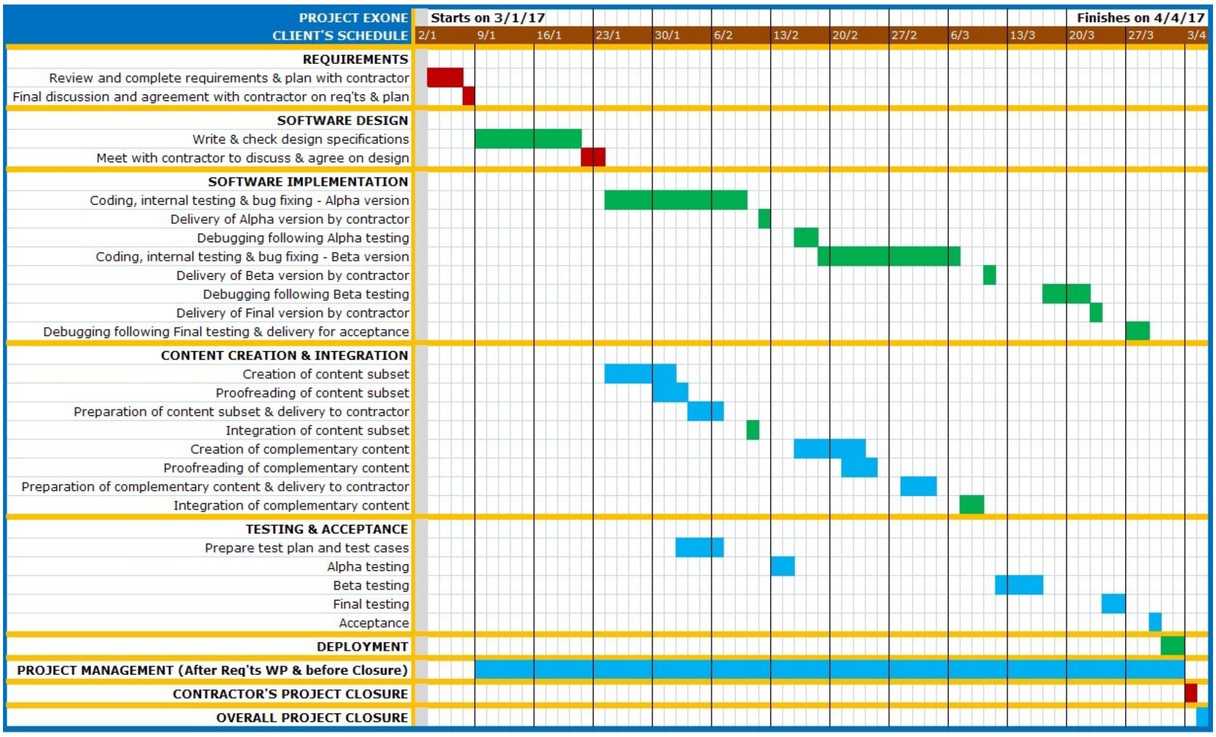
Tools such as MS Project (or ProjectLibre) can display schedules in various forms, but they are not always suited to presentations or inclusion in reports because they may extend over several pages.

I strongly recommend creating a **single-page overview** of the schedule for project supervision and reporting purposes, in addition to the schedule produced with a project management application, which may be difficult to read and understand at a glance.

- Example: Contractor's schedule for project EXONE created with a spreadsheet tool:



➢ Example: Client's schedule for project EXONE created with a spreadsheet tool:



PROJECT EXONE
CLIENT'S SCHEDULE

Starts on 3/1/17 — Finishes on 4/4/17

2/1 | 9/1 | 16/1 | 23/1 | 30/1 | 6/2 | 13/2 | 20/2 | 27/2 | 6/3 | 13/3 | 20/3 | 27/3 | 3/4

REQUIREMENTS
Review and complete requirements & plan with contractor
Final discussion and agreement with contractor on req'ts & plan

SOFTWARE DESIGN
Write & check design specifications
Meet with contractor to discuss & agree on design

SOFTWARE IMPLEMENTATION
Coding, internal testing & bug fixing - Alpha version
Delivery of Alpha version by contractor
Debugging following Alpha testing
Coding, internal testing & bug fixing - Beta version
Delivery of Beta version by contractor
Debugging following Beta testing
Delivery of Final version by contractor
Debugging following Final testing & delivery for acceptance

CONTENT CREATION & INTEGRATION
Creation of content subset
Proofreading of content subset
Preparation of content subset & delivery to contractor
Integration of content subset
Creation of complementary content
Proofreading of complementary content
Preparation of complementary content & delivery to contractor
Integration of complementary content

TESTING & ACCEPTANCE
Prepare test plan and test cases
Alpha testing
Beta testing
Final testing
Acceptance

DEPLOYMENT

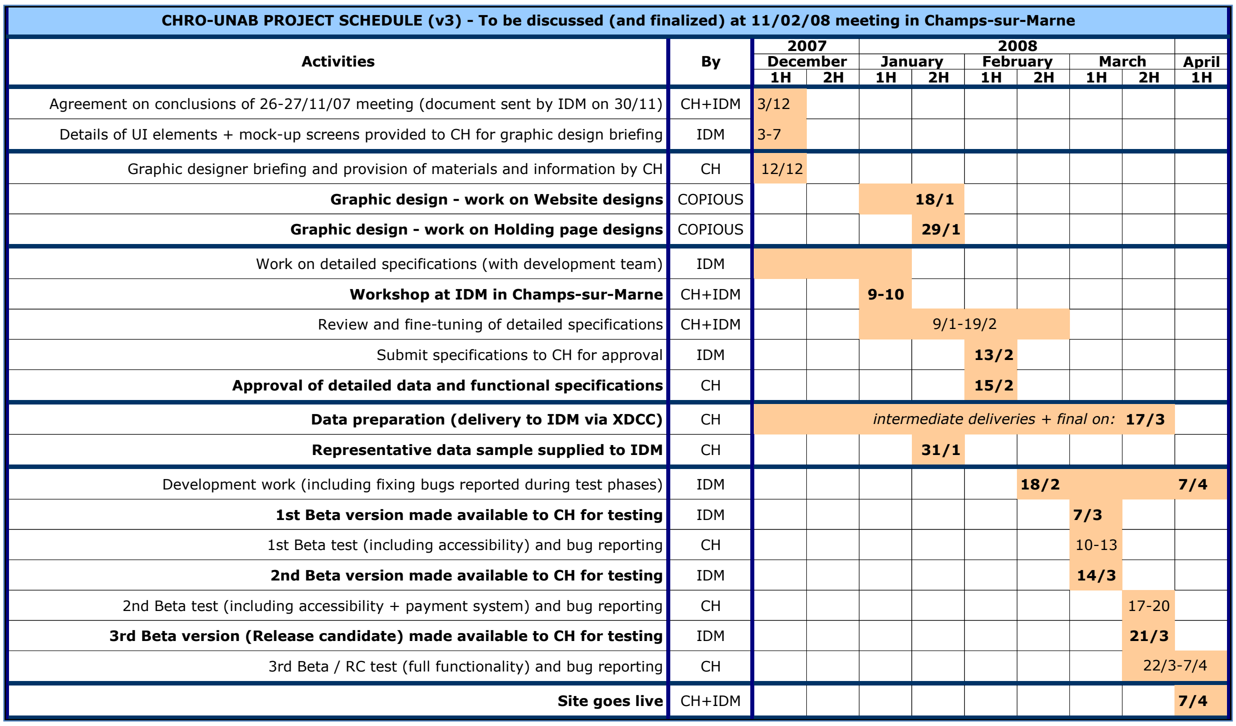
PROJECT MANAGEMENT (After Req'ts WP & before Closure)

CONTRACTOR'S PROJECT CLOSURE

OVERALL PROJECT CLOSURE

In these two EXONE schedules, colour codes are used to show whose resources are involved in the various tasks/WPs: one colour for tasks/WPs performed by contractor's resources only, one colour for tasks/WPs performed by client's resources only, and a third colour for tasks/WPs involving both contractor's resources and client's resources.

➢ Other example of a summary schedule in spreadsheet format:



CHRO-UNAB PROJECT SCHEDULE (v3) - To be discussed (and finalized) at 11/02/08 meeting in Champs-sur-Marne

Activities	By	2007		2008						
		December		January		February		March		April
		1H	2H	1H	2H	1H	2H	1H	2H	1H
Agreement on conclusions of 26-27/11/07 meeting (document sent by IDM on 30/11)	CH+IDM	3/12								
Details of UI elements + mock-up screens provided to CH for graphic design briefing	IDM	3-7								
Graphic designer briefing and provision of materials and information by CH	CH	12/12								
Graphic design - work on Website designs	COPIOUS				**18/1**					
Graphic design - work on Holding page designs	COPIOUS				**29/1**					
Work on detailed specifications (with development team)	IDM									
Workshop at IDM in Champs-sur-Marne	CH+IDM			**9-10**						
Review and fine-tuning of detailed specifications	CH+IDM				9/1-19/2					
Submit specifications to CH for approval	IDM					**13/2**				
Approval of detailed data and functional specifications	CH					**15/2**				
Data preparation (delivery to IDM via XDCC)	CH			*intermediate deliveries + final on:*					**17/3**	
Representative data sample supplied to IDM	CH				**31/1**					
Development work (including fixing bugs reported during test phases)	IDM						**18/2**			**7/4**
1st Beta version made available to CH for testing	IDM							**7/3**		
1st Beta test (including accessibility) and bug reporting	CH							10-13		
2nd Beta version made available to CH for testing	IDM							**14/3**		
2nd Beta test (including accessibility + payment system) and bug reporting	CH								17-20	
3rd Beta version (Release candidate) made available to CH for testing	IDM								**21/3**	
3rd Beta / RC test (full functionality) and bug reporting	CH								22/3-7/4	
Site goes live	CH+IDM									**7/4**

The above schedule (for a "real-life" online dictionaries project, which I was involved in as a freelance PM) features milestone dates and a Resources ("By") column, where "CH" refers to the client (Chambers Harrap, the publisher), "IDM" is the software development contractor and "COPIOUS" is the digital agency used by CH for graphic design work.

Estimate costs

This process consists in estimating the **cost of resources** needed to successfully complete the project tasks and therefore the project itself.

Costs are expressed in numbers of **monetary units** (euros, pounds, dollars...).
Human resource costs may be initially measured in numbers of **work period units** (hours, days…) but of course they will eventually need to be converted into monetary units in order to establish the budget of the project.

Cost estimates may be refined as the project moves forward and more details become available.

The result of the cost estimates, as well as the budget, may be presented in various formats, usually in a spreadsheet or in a document produced with a project management application (eg MS Project or ProjectLibre). The use of **standard templates** may be required in certain organizations or circumstances.
The detailed **estimates may be summarized in a cost document** showing costs per type of resource and per task or work package.

In the case of projects that extend over a long period, say more than a year, **inflation** may need to be taken into consideration, in particular for labour costs (salaries may indeed evolve more or less in proportion to inflation).

Likewise, any foreseeable increase or decrease of the **cost of materials**, if any, should be factored in (eg hardware may cost less in year 2 than it does in year 1, so the total amount of a project's **bill of materials** ("BOM") may be significantly reduced).

The cost document should also include all **assumptions** that have been made, as well as known **constraints and risks** that may have an impact on costs.

Certain costs may incorporate a "**contingency reserve**" (or "**contingency allowance**") to take into account any changes that may result from the realization of potential risks identified in the planning process. The reserve may be a percentage of the initial cost estimate or a fixed amount. Contingency reserves should be clearly documented.

Below is a **non-exhaustive list of cost elements** to be taken into consideration.

- **Labour**, for which costs are derived on a pro-rata basis from gross salaries, bonuses, social contributions paid by the employer, perquisites (benefits in kind) and possible end-of-project awards.
- **Overhead**, generally expressed as a percentage of gross labour costs, corresponding to a share of the cost of company-wide or department-wide services such as HR, Finance, Facilities, Marketing, General Management, etc.
- **Hiring** (job posting, recruitment agency services, etc.).
- **Offices** (building, floor space, furniture, utilities including communications), if not already included in the overhead costs.
- **Workstations** (computers and software licences).
- **Servers** (hardware and software licences).
- **Other equipment** (printers, scanners, audiovisual equipment, tools, etc.).
- **Supplies/consumables** (paper, pens, ink cartridges, CD/DVD-ROMs, etc.).
- **Materials** (eg hardware that may be required to build a prototype).
- **Documentation** (reference books, subscriptions to online services, etc.).

- **Competitors’ products** (software, online service, etc.).
- **Multimedia assets** (acquisition of rights for elements of content of a product).
- **Travel** (transport, accommodation, food).
- **Events** (eg special, exceptional off-site meetings: venue rental, breaks, meals, etc.).
- **Services** (staff training, brainstorming coach/facilitator, hosting service, etc.).
- **Consultancy** (specific expertise, legal advice and support, accounts auditing, etc.).
- **Contractors** (writing, proofreading, translation, graphic design, software development, testing, etc.).

It may make sense to rent equipment (eg computers) for the duration of certain tasks (eg testing) and/or for the duration of temporary staff contracts. A “**rent or buy**” evaluation should therefore be made as part of the cost estimation process.
Likewise, it may be more cost-effective to **subcontract** certain tasks (eg graphic design, testing) than to perform them in-house. A “**make or buy**” evaluation should therefore be made for such tasks.

The PM may need help from a financial controller to estimate costs. For example, “**overhead**”, as mentioned in the above list, is often a percentage of gross labour costs, such percentage being calculated and provided by the Finance department (or the Human Resources department).
However, the PM usually needs to do much of the estimating himself, using whatever sources of information are available, which may require a fair amount of research.

The following spreadsheet provides an **example** of the rough calculation of the **cost (in euros) per day of a salaried employee in France**, taking into account the gross annual salary, the employer’s social contributions, a standard overhead rate and the actual number of working days in a year. The cost per day is calculated by dividing the total annual cost by the number of actual workdays in a year.

Weeks per year	52
Days per year excluding weekends	260
Public holidays	10
Total potential workdays	250
Vacation days	30
Total actual workdays in a year	**220**
Gross annual salary	**36,000**
Employer's social contributions (%)	50%
Employer's social contributions	18,000
Annual cost excluding overhead	54,000
Overhead (%)	33%
Overhead	17,820
Annual cost including overhead	71,820
Total cost / gross salary	*2*
Cost per day of work	**326**

The part of an employee’s **cost that cannot be allocated directly to a project** (or several projects) over a given period needs to be included in the organization’s overhead. It is also common practice to take into account “**idle time**”, ie a fraction of the standard number of work hours during which no work is actually done (breaks, etc.), so the theoretical cost per day of work needs to be increased by some percentage, eg 10%.

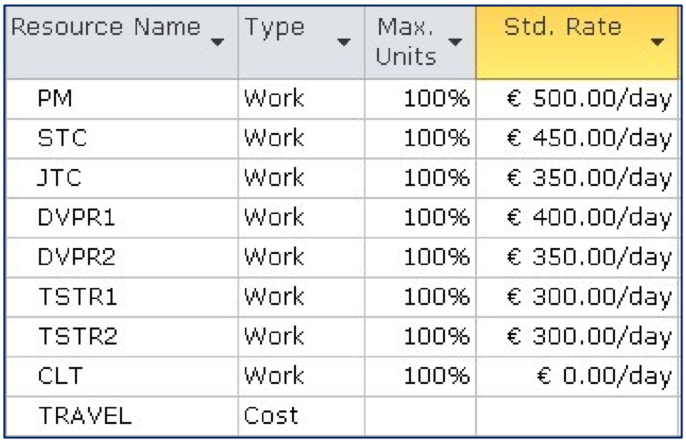
In France, a **quick estimate of cost per day** including approximately 9% idle time can be obtained **by dividing the gross annual salary by 100**.

The **cost of contractors** (eg freelances) is often calculated on the basis of a fixed amount per day, but it may be a flat fee negotiated for a given period (the average cost per day hopefully being lower than the "standard" cost as a result of the negotiation…).

Note that contractor fees as well as other costs are always quoted "excluding tax". Value-added tax ("VAT") does not need to be taken into account for the estimation of project costs, except if VAT cannot be reclaimed (if in doubt, check with your Accounting/Finance department…).

Converting human resource costs expressed in work period units into costs expressed in monetary units is done automatically by project management applications based on workload information (eg number of person-days per task) and on monetary cost information (eg amount per resource per day) provided in a "**Resource sheet**", as shown in the following illustrations for project EXONE.

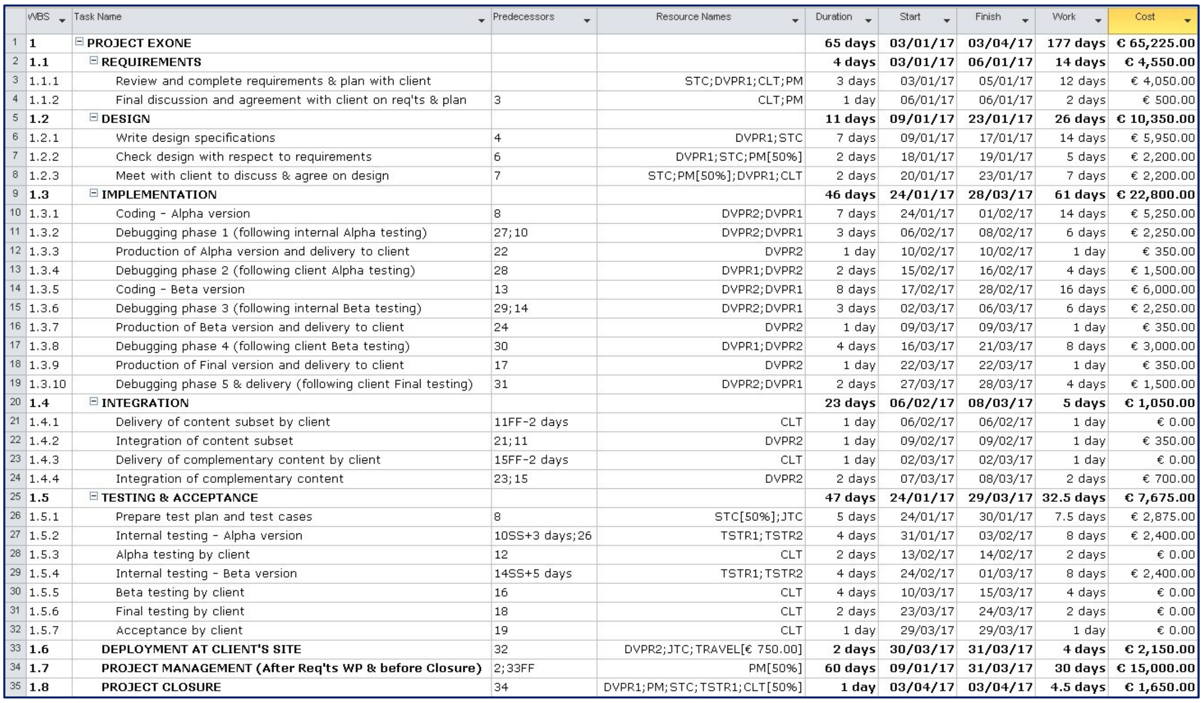
- Project EXONE – Contractor's resource sheet:



Resource Name	Type	Max. Units	Std. Rate
PM	Work	100%	€ 500.00/day
STC	Work	100%	€ 450.00/day
JTC	Work	100%	€ 350.00/day
DVPR1	Work	100%	€ 400.00/day
DVPR2	Work	100%	€ 350.00/day
TSTR1	Work	100%	€ 300.00/day
TSTR2	Work	100%	€ 300.00/day
CLT	Work	100%	€ 0.00/day
TRAVEL	Cost		

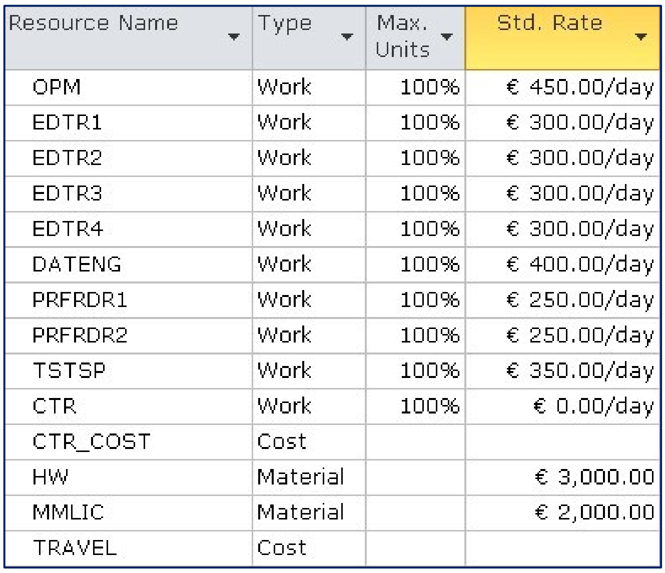
Note that the "Standard Rate" for the client (CLT) has been set to 0 in this example in order to restrict the calculation of costs to the contractor's resources and work.

- Project EXONE – Gantt table view featuring the contractor's costs:



	WBS	Task Name	Predecessors	Resource Names	Duration	Start	Finish	Work	Cost
1	**1**	⊟ **PROJECT EXONE**			**65 days**	**03/01/17**	**03/04/17**	**177 days**	**€ 65,225.00**
2	**1.1**	⊟ **REQUIREMENTS**			**4 days**	**03/01/17**	**06/01/17**	**14 days**	**€ 4,550.00**
3	1.1.1	Review and complete requirements & plan with client		STC;DVPR1;CLT;PM	3 days	03/01/17	05/01/17	12 days	€ 4,050.00
4	1.1.2	Final discussion and agreement with client on req'ts & plan	3	CLT;PM	1 day	06/01/17	06/01/17	2 days	€ 500.00
5	**1.2**	⊟ **DESIGN**			**11 days**	**09/01/17**	**23/01/17**	**26 days**	**€ 10,350.00**
6	1.2.1	Write design specifications	4	DVPR1;STC	7 days	09/01/17	17/01/17	14 days	€ 5,950.00
7	1.2.2	Check design with respect to requirements	6	DVPR1;STC;PM[50%]	2 days	18/01/17	19/01/17	5 days	€ 2,200.00
8	1.2.3	Meet with client to discuss & agree on design	7	STC;PM[50%];DVPR1;CLT	2 days	20/01/17	23/01/17	7 days	€ 2,200.00
9	**1.3**	⊟ **IMPLEMENTATION**			**46 days**	**24/01/17**	**28/03/17**	**61 days**	**€ 22,800.00**
10	1.3.1	Coding - Alpha version	8	DVPR2;DVPR1	7 days	24/01/17	01/02/17	14 days	€ 5,250.00
11	1.3.2	Debugging phase 1 (following internal Alpha testing)	27;10	DVPR2;DVPR1	3 days	06/02/17	08/02/17	6 days	€ 2,250.00
12	1.3.3	Production of Alpha version and delivery to client	22	DVPR2	1 day	10/02/17	10/02/17	1 day	€ 350.00
13	1.3.4	Debugging phase 2 (following client Alpha testing)	28	DVPR1;DVPR2	2 days	15/02/17	16/02/17	4 days	€ 1,500.00
14	1.3.5	Coding - Beta version	13	DVPR2;DVPR1	8 days	17/02/17	28/02/17	16 days	€ 6,000.00
15	1.3.6	Debugging phase 3 (following internal Beta testing)	29;14	DVPR2;DVPR1	3 days	02/03/17	06/03/17	6 days	€ 2,250.00
16	1.3.7	Production of Beta version and delivery to client	24	DVPR2	1 day	09/03/17	09/03/17	1 day	€ 350.00
17	1.3.8	Debugging phase 4 (following client Beta testing)	30	DVPR1;DVPR2	4 days	16/03/17	21/03/17	8 days	€ 3,000.00
18	1.3.9	Production of Final version and delivery to client	17	DVPR2	1 day	22/03/17	22/03/17	1 day	€ 350.00
19	1.3.10	Debugging phase 5 & delivery (following client Final testing)	31	DVPR2;DVPR1	2 days	27/03/17	28/03/17	4 days	€ 1,500.00
20	**1.4**	⊟ **INTEGRATION**			**23 days**	**06/02/17**	**08/03/17**	**5 days**	**€ 1,050.00**
21	1.4.1	Delivery of content subset by client	11FF-2 days	CLT	1 day	06/02/17	06/02/17	1 day	€ 0.00
22	1.4.2	Integration of content subset	21;11	DVPR2	1 day	09/02/17	09/02/17	1 day	€ 350.00
23	1.4.3	Delivery of complementary content by client	15FF-2 days	CLT	1 day	02/03/17	02/03/17	1 day	€ 0.00
24	1.4.4	Integration of complementary content	23;15	DVPR2	2 days	07/03/17	08/03/17	2 days	€ 700.00
25	**1.5**	⊟ **TESTING & ACCEPTANCE**			**47 days**	**24/01/17**	**29/03/17**	**32.5 days**	**€ 7,675.00**
26	1.5.1	Prepare test plan and test cases	8	STC[50%];JTC	5 days	24/01/17	30/01/17	7.5 days	€ 2,875.00
27	1.5.2	Internal testing - Alpha version	10SS+3 days;26	TSTR1;TSTR2	4 days	31/01/17	03/02/17	8 days	€ 2,400.00
28	1.5.3	Alpha testing by client	12	CLT	2 days	13/02/17	14/02/17	2 days	€ 0.00
29	1.5.4	Internal testing - Beta version	14SS+5 days	TSTR1;TSTR2	4 days	24/02/17	01/03/17	8 days	€ 2,400.00
30	1.5.5	Beta testing by client	16	CLT	4 days	10/03/17	15/03/17	4 days	€ 0.00
31	1.5.6	Final testing by client	18	CLT	2 days	23/03/17	24/03/17	2 days	€ 0.00
32	1.5.7	Acceptance by client	19	CLT	1 day	29/03/17	29/03/17	1 day	€ 0.00
33	**1.6**	**DEPLOYMENT AT CLIENT'S SITE**	32	DVPR2;JTC;TRAVEL[€ 750.00]	**2 days**	**30/03/17**	**31/03/17**	**4 days**	**€ 2,150.00**
34	**1.7**	**PROJECT MANAGEMENT (After Req'ts WP & before Closure)**	2;33FF	PM[50%]	**60 days**	**09/01/17**	**31/03/17**	**30 days**	**€ 15,000.00**
35	**1.8**	**PROJECT CLOSURE**	34	DVPR1;PM;STC;TSTR1;CLT[50%]	**1 day**	**03/04/17**	**03/04/17**	**4.5 days**	**€ 1,650.00**

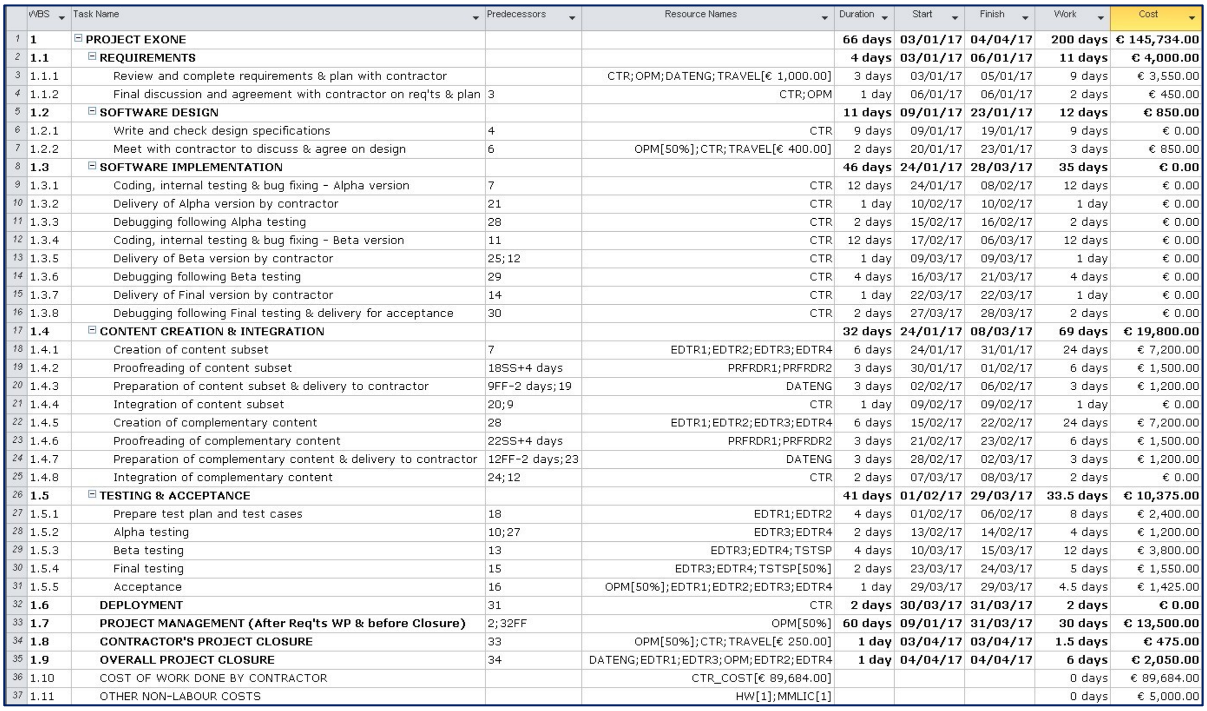
- Project EXONE – Client's resource sheet:



Resource Name	Type	Max. Units	Std. Rate
OPM	Work	100%	€ 450.00/day
EDTR1	Work	100%	€ 300.00/day
EDTR2	Work	100%	€ 300.00/day
EDTR3	Work	100%	€ 300.00/day
EDTR4	Work	100%	€ 300.00/day
DATENG	Work	100%	€ 400.00/day
PRFRDR1	Work	100%	€ 250.00/day
PRFRDR2	Work	100%	€ 250.00/day
TSTSP	Work	100%	€ 350.00/day
CTR	Work	100%	€ 0.00/day
CTR_COST	Cost		
HW	Material		€ 3,000.00
MMLIC	Material		€ 2,000.00
TRAVEL	Cost		

Note that the “Standard Rate” for the contractor (CTR) has been set to 0 in this example in order to restrict the calculation of costs to the client’s resources and work.

- Project EXONE – Gantt table view featuring the client’s costs:

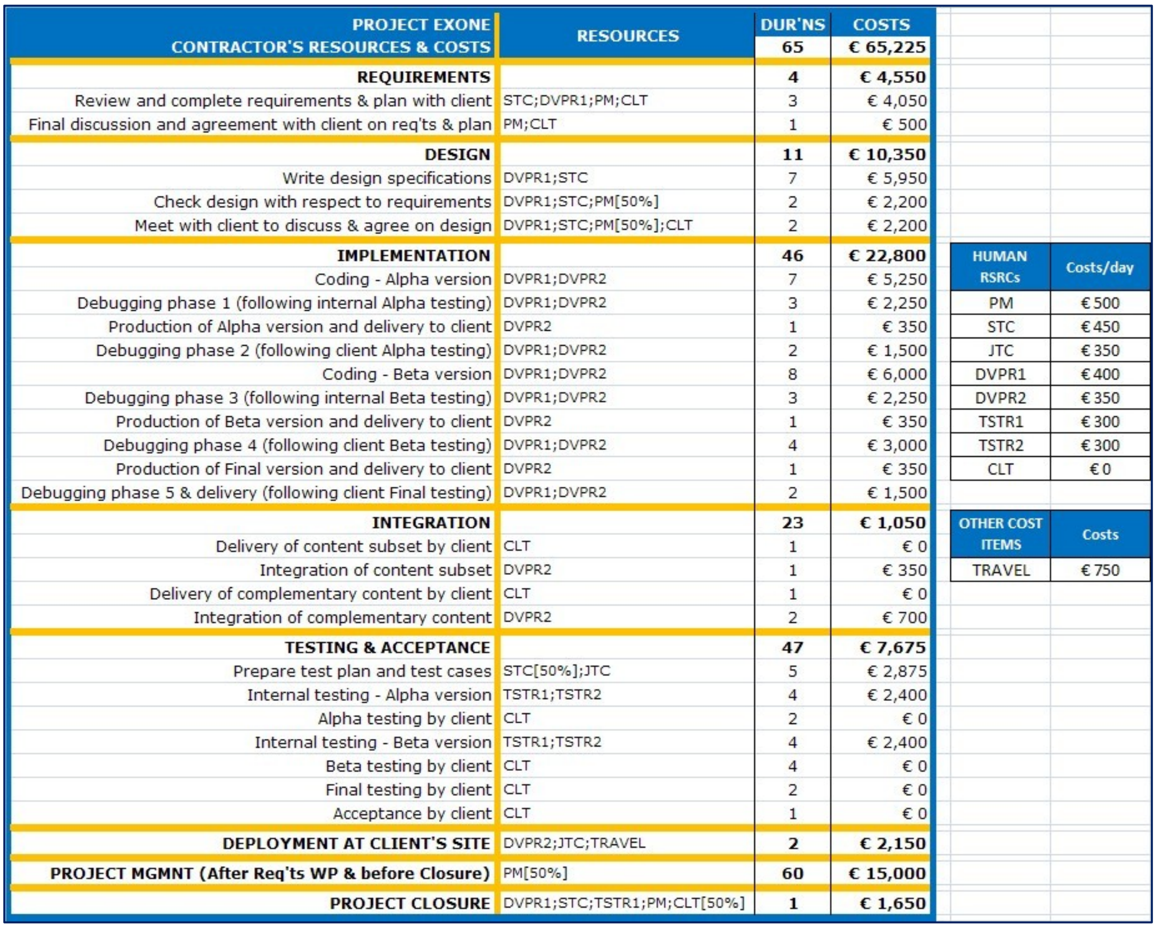


	WBS	Task Name	Predecessors	Resource Names	Duration	Start	Finish	Work	Cost
1	**1**	⊟ **PROJECT EXONE**			**66 days**	**03/01/17**	**04/04/17**	**200 days**	**€ 145,734.00**
2	**1.1**	⊟ **REQUIREMENTS**			**4 days**	**03/01/17**	**06/01/17**	**11 days**	**€ 4,000.00**
3	1.1.1	Review and complete requirements & plan with contractor		CTR;OPM;DATENG;TRAVEL[€ 1,000.00]	3 days	03/01/17	05/01/17	9 days	€ 3,550.00
4	1.1.2	Final discussion and agreement with contractor on req'ts & plan	3	CTR;OPM	1 day	06/01/17	06/01/17	2 days	€ 450.00
5	**1.2**	⊟ **SOFTWARE DESIGN**			**11 days**	**09/01/17**	**23/01/17**	**12 days**	**€ 850.00**
6	1.2.1	Write and check design specifications	4	CTR	9 days	09/01/17	19/01/17	9 days	€ 0.00
7	1.2.2	Meet with contractor to discuss & agree on design	6	OPM[50%];CTR;TRAVEL[€ 400.00]	2 days	20/01/17	23/01/17	3 days	€ 850.00
8	**1.3**	⊟ **SOFTWARE IMPLEMENTATION**			**46 days**	**24/01/17**	**28/03/17**	**35 days**	**€ 0.00**
9	1.3.1	Coding, internal testing & bug fixing - Alpha version	7	CTR	12 days	24/01/17	08/02/17	12 days	€ 0.00
10	1.3.2	Delivery of Alpha version by contractor	21	CTR	1 day	10/02/17	10/02/17	1 day	€ 0.00
11	1.3.3	Debugging following Alpha testing	28	CTR	2 days	15/02/17	16/02/17	2 days	€ 0.00
12	1.3.4	Coding, internal testing & bug fixing - Beta version	11	CTR	12 days	17/02/17	06/03/17	12 days	€ 0.00
13	1.3.5	Delivery of Beta version by contractor	25;12	CTR	1 day	09/03/17	09/03/17	1 day	€ 0.00
14	1.3.6	Debugging following Beta testing	29	CTR	4 days	16/03/17	21/03/17	4 days	€ 0.00
15	1.3.7	Delivery of Final version by contractor	14	CTR	1 day	22/03/17	22/03/17	1 day	€ 0.00
16	1.3.8	Debugging following Final testing & delivery for acceptance	30	CTR	2 days	27/03/17	28/03/17	2 days	€ 0.00
17	**1.4**	⊟ **CONTENT CREATION & INTEGRATION**			**32 days**	**24/01/17**	**08/03/17**	**69 days**	**€ 19,800.00**
18	1.4.1	Creation of content subset	7	EDTR1;EDTR2;EDTR3;EDTR4	6 days	24/01/17	31/01/17	24 days	€ 7,200.00
19	1.4.2	Proofreading of content subset	18SS+4 days	PRFRDR1;PRFRDR2	3 days	30/01/17	01/02/17	6 days	€ 1,500.00
20	1.4.3	Preparation of content subset & delivery to contractor	9FF-2 days;19	DATENG	3 days	02/02/17	06/02/17	3 days	€ 1,200.00
21	1.4.4	Integration of content subset	20;9	CTR	1 day	09/02/17	09/02/17	1 day	€ 0.00
22	1.4.5	Creation of complementary content	28	EDTR1;EDTR2;EDTR3;EDTR4	6 days	15/02/17	22/02/17	24 days	€ 7,200.00
23	1.4.6	Proofreading of complementary content	22SS+4 days	PRFRDR1;PRFRDR2	3 days	21/02/17	23/02/17	6 days	€ 1,500.00
24	1.4.7	Preparation of complementary content & delivery to contractor	12FF-2 days;23	DATENG	3 days	28/02/17	02/03/17	3 days	€ 1,200.00
25	1.4.8	Integration of complementary content	24;12	CTR	2 days	07/03/17	08/03/17	2 days	€ 0.00
26	**1.5**	⊟ **TESTING & ACCEPTANCE**			**41 days**	**01/02/17**	**29/03/17**	**33.5 days**	**€ 10,375.00**
27	1.5.1	Prepare test plan and test cases	18	EDTR1;EDTR2	4 days	01/02/17	06/02/17	8 days	€ 2,400.00
28	1.5.2	Alpha testing	10;27	EDTR3;EDTR4	2 days	13/02/17	14/02/17	4 days	€ 1,200.00
29	1.5.3	Beta testing	13	EDTR3;EDTR4;TSTSP	4 days	10/03/17	15/03/17	12 days	€ 3,800.00
30	1.5.4	Final testing	15	EDTR3;EDTR4;TSTSP[50%]	2 days	23/03/17	24/03/17	5 days	€ 1,550.00
31	1.5.5	Acceptance	16	OPM[50%];EDTR1;EDTR2;EDTR3;EDTR4	1 day	29/03/17	29/03/17	4.5 days	€ 1,425.00
32	**1.6**	**DEPLOYMENT**	31	CTR	**2 days**	**30/03/17**	**31/03/17**	**2 days**	**€ 0.00**
33	**1.7**	**PROJECT MANAGEMENT (After Req'ts WP & before Closure)**	2;32FF	OPM[50%]	**60 days**	**09/01/17**	**31/03/17**	**30 days**	**€ 13,500.00**
34	**1.8**	**CONTRACTOR'S PROJECT CLOSURE**	33	OPM[50%];CTR;TRAVEL[€ 250.00]	**1 day**	**03/04/17**	**03/04/17**	**1.5 days**	**€ 475.00**
35	**1.9**	**OVERALL PROJECT CLOSURE**	34	DATENG;EDTR1;EDTR3;OPM;EDTR2;EDTR4	**1 day**	**04/04/17**	**04/04/17**	**6 days**	**€ 2,050.00**
36	1.10	COST OF WORK DONE BY CONTRACTOR		CTR_COST[€ 89,684.00]				0 days	€ 89,684.00
37	1.11	OTHER NON-LABOUR COSTS		HW[1];MMLIC[1]				0 days	€ 5,000.00

As illustrated above, **non-labour resources and corresponding costs** can be featured in a plan produced with an application such as MS Project (or ProjectLibre).
With MS Project, resources of type "**Material**", such as "HW" (for hardware), are given a unit value in the Resource sheet and a number of units in the Gantt view of the plan, whereas resources of type "**Cost**", such as "TRAVEL", are not given any value in the Resource sheet, so a value needs to be assigned to them in the Gantt view of the plan so that their costs are included in the total cost of the project.

In the above example, "TRAVEL" has been assigned to certain "real" tasks, whereas other non-labour resources have been assigned to "dummy" tasks listed after "OVERALL PROJECT CLOSURE", in particular "CTR_COST", whose value is the price quoted by the contractor for the work done for the client on project EXONE (as explained further on).

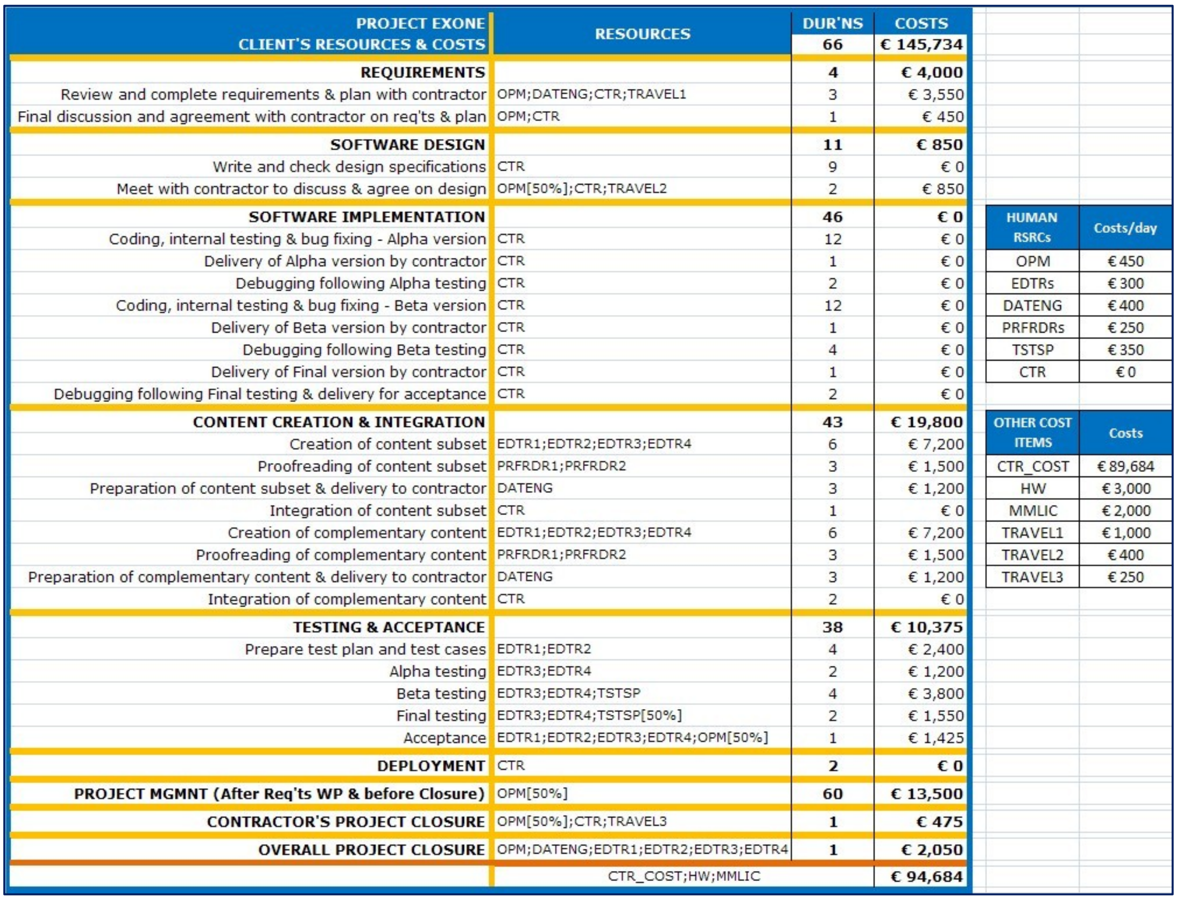
The following illustrations are examples of how to feature the costs of resources in a plan produced with a spreadsheet application.



PROJECT EXONE	RESOURCES	DUR'NS	COSTS
CONTRACTOR'S RESOURCES & COSTS		65	€ 65,225
REQUIREMENTS		**4**	**€ 4,550**
Review and complete requirements & plan with client	STC;DVPR1;PM;CLT	3	€ 4,050
Final discussion and agreement with client on req'ts & plan	PM;CLT	1	€ 500
DESIGN		**11**	**€ 10,350**
Write design specifications	DVPR1;STC	7	€ 5,950
Check design with respect to requirements	DVPR1;STC;PM[50%]	2	€ 2,200
Meet with client to discuss & agree on design	DVPR1;STC;PM[50%];CLT	2	€ 2,200
IMPLEMENTATION		**46**	**€ 22,800**
Coding - Alpha version	DVPR1;DVPR2	7	€ 5,250
Debugging phase 1 (following internal Alpha testing)	DVPR1;DVPR2	3	€ 2,250
Production of Alpha version and delivery to client	DVPR2	1	€ 350
Debugging phase 2 (following client Alpha testing)	DVPR1;DVPR2	2	€ 1,500
Coding - Beta version	DVPR1;DVPR2	8	€ 6,000
Debugging phase 3 (following internal Beta testing)	DVPR1;DVPR2	3	€ 2,250
Production of Beta version and delivery to client	DVPR2	1	€ 350
Debugging phase 4 (following client Beta testing)	DVPR1;DVPR2	4	€ 3,000
Production of Final version and delivery to client	DVPR2	1	€ 350
Debugging phase 5 & delivery (following client Final testing)	DVPR1;DVPR2	2	€ 1,500
INTEGRATION		**23**	**€ 1,050**
Delivery of content subset by client	CLT	1	€ 0
Integration of content subset	DVPR2	1	€ 350
Delivery of complementary content by client	CLT	1	€ 0
Integration of complementary content	DVPR2	2	€ 700
TESTING & ACCEPTANCE		**47**	**€ 7,675**
Prepare test plan and test cases	STC[50%];JTC	5	€ 2,875
Internal testing - Alpha version	TSTR1;TSTR2	4	€ 2,400
Alpha testing by client	CLT	2	€ 0
Internal testing - Beta version	TSTR1;TSTR2	4	€ 2,400
Beta testing by client	CLT	4	€ 0
Final testing by client	CLT	2	€ 0
Acceptance by client	CLT	1	€ 0
DEPLOYMENT AT CLIENT'S SITE	DVPR2;JTC;TRAVEL	**2**	**€ 2,150**
PROJECT MGMNT (After Req'ts WP & before Closure)	PM[50%]	**60**	**€ 15,000**
PROJECT CLOSURE	DVPR1;STC;TSTR1;PM;CLT[50%]	**1**	**€ 1,650**

HUMAN RSRCs	Costs/day
PM	€ 500
STC	€ 450
JTC	€ 350
DVPR1	€ 400
DVPR2	€ 350
TSTR1	€ 300
TSTR2	€ 300
CLT	€ 0

OTHER COST ITEMS	Costs
TRAVEL	€ 750



PROJECT EXONE	RESOURCES	DUR'NS	COSTS
CLIENT'S RESOURCES & COSTS		66	€ 145,734
REQUIREMENTS		**4**	**€ 4,000**
Review and complete requirements & plan with contractor	OPM;DATENG;CTR;TRAVEL1	3	€ 3,550
Final discussion and agreement with contractor on req'ts & plan	OPM;CTR	1	€ 450
SOFTWARE DESIGN		**11**	**€ 850**
Write and check design specifications	CTR	9	€ 0
Meet with contractor to discuss & agree on design	OPM[50%];CTR;TRAVEL2	2	€ 850
SOFTWARE IMPLEMENTATION		**46**	**€ 0**
Coding, internal testing & bug fixing - Alpha version	CTR	12	€ 0
Delivery of Alpha version by contractor	CTR	1	€ 0
Debugging following Alpha testing	CTR	2	€ 0
Coding, internal testing & bug fixing - Beta version	CTR	12	€ 0
Delivery of Beta version by contractor	CTR	1	€ 0
Debugging following Beta testing	CTR	4	€ 0
Delivery of Final version by contractor	CTR	1	€ 0
Debugging following Final testing & delivery for acceptance	CTR	2	€ 0
CONTENT CREATION & INTEGRATION		**43**	**€ 19,800**
Creation of content subset	EDTR1;EDTR2;EDTR3;EDTR4	6	€ 7,200
Proofreading of content subset	PRFRDR1;PRFRDR2	3	€ 1,500
Preparation of content subset & delivery to contractor	DATENG	3	€ 1,200
Integration of content subset	CTR	1	€ 0
Creation of complementary content	EDTR1;EDTR2;EDTR3;EDTR4	6	€ 7,200
Proofreading of complementary content	PRFRDR1;PRFRDR2	3	€ 1,500
Preparation of complementary content & delivery to contractor	DATENG	3	€ 1,200
Integration of complementary content	CTR	2	€ 0
TESTING & ACCEPTANCE		**38**	**€ 10,375**
Prepare test plan and test cases	EDTR1;EDTR2	4	€ 2,400
Alpha testing	EDTR3;EDTR4	2	€ 1,200
Beta testing	EDTR3;EDTR4;TSTSP	4	€ 3,800
Final testing	EDTR3;EDTR4;TSTSP[50%]	2	€ 1,550
Acceptance	EDTR1;EDTR2;EDTR3;EDTR4;OPM[50%]	1	€ 1,425
DEPLOYMENT	CTR	**2**	**€ 0**
PROJECT MGMNT (After Req'ts WP & before Closure)	OPM[50%]	**60**	**€ 13,500**
CONTRACTOR'S PROJECT CLOSURE	OPM[50%];CTR;TRAVEL3	**1**	**€ 475**
OVERALL PROJECT CLOSURE	OPM;DATENG;EDTR1;EDTR2;EDTR3;EDTR4	**1**	**€ 2,050**
	CTR_COST;HW;MMLIC		**€ 94,684**

HUMAN RSRCs	Costs/day
OPM	€ 450
EDTRs	€ 300
DATENG	€ 400
PRFRDRs	€ 250
TSTSP	€ 350
CTR	€ 0

OTHER COST ITEMS	Costs
CTR_COST	€ 89,684
HW	€ 3,000
MMLIC	€ 2,000
TRAVEL1	€ 1,000
TRAVEL2	€ 400
TRAVEL3	€ 250

Determine the budget

The budget of the project is determined by **adding up all estimated costs**.

The budget, which represents the total amount of funds required for the project to be successfully completed, should be officially **approved by the project sponsor**.

If the budget amounts to more than had been authorized in the project charter, the sponsor may approve the corresponding extension to the budget or request that it should be aligned with the initial estimate. In the latter case, the project plan will need to be reworked.

Individual task costs generally do not require formal approval. In some cases however, the budget for work packages with a significant cost may require such an approval.

The **authorized budget** is the **baseline** for the evaluation of the project's "**cost performance**".

In addition to the contingency reserves that may have been included in the cost estimates, the budget may incorporate a "**management reserve**", ie a percentage or a lump sum added to the total cost estimate to provide for any possible cost increase due to "**imponderables**". The management reserve should be clearly documented.

If **funding** can be (or has to be) **staged**, a corresponding view of the budget should be provided in order to show the funding requirements for each stage of the project. A stage may be a phase of the project or a time period (eg month, quarter, year).

As mentioned in chapter 6 ("P&L"), a budget (or price) established by a contractor (or subcontractor) for the execution of part of a client's project generally includes a **profit margin**. The absolute value of the profit margin is added to the total cost of the work executed by the contractor.

When expressed as a percentage, the margin is calculated by convention in relation to the price of the work, not to its cost.

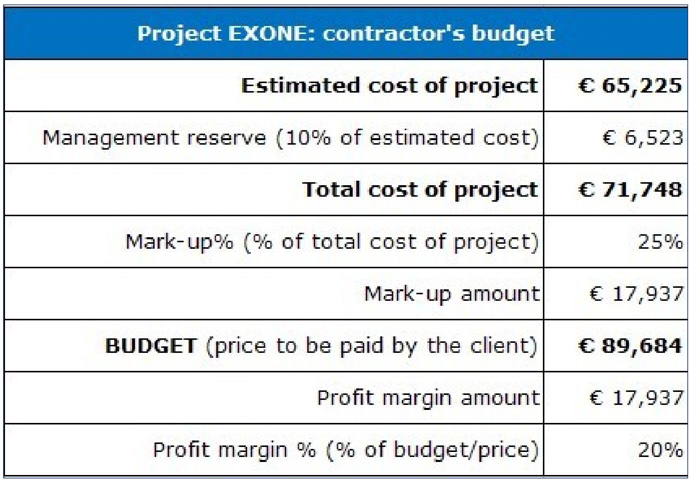
As for the mark-up percentage, it is calculated in relation to the cost of the work, not to its price. Of course, the absolute values of the margin and the mark-up are identical. (For details, return to chapter 6, section "Margin vs Mark-up".)

For example, if the total cost is estimated at 100,000 euros and the required profit margin is 20%, the price should amount to 125,000 euros, the result of 100,000/0.8, as per the following formula: Price = Cost / (1 - Margin%).

If that price is accepted, the client will pay 125,000 euros to the contractor, which will cover the contractor's cost of 100,000 euros and provide the contractor with a profit margin of 25,000 euros (which is what was required, ie 20% of 125,000 euros).
In other words, the contractor applies a 25% "mark-up" to costs in order to include a 20% profit margin in the price to be paid by the client (100,000 x 1.25 = 125,000).

Below are two spreadsheets that provide **examples of summary budgets**.

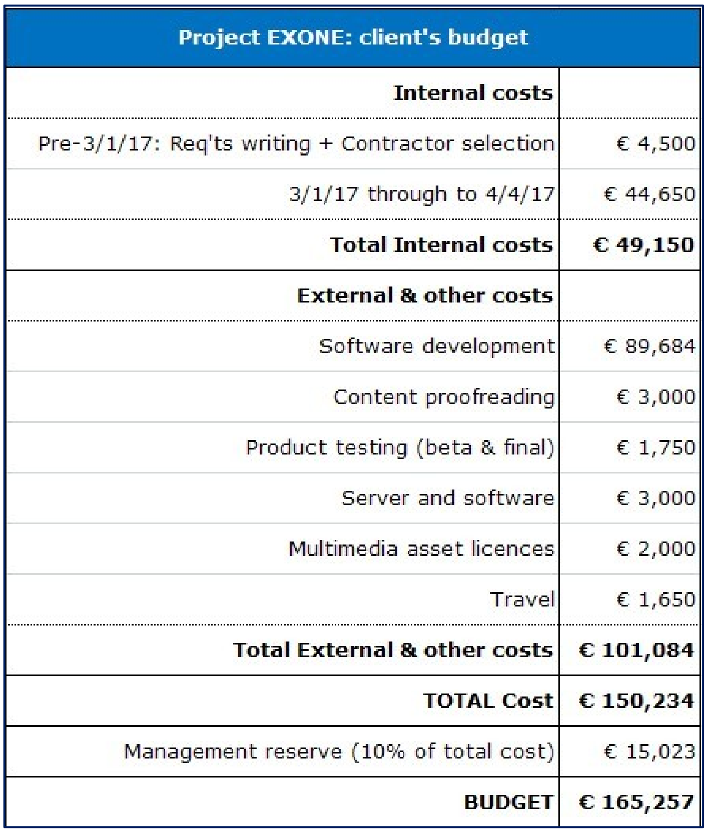
- Project EXONE – Contractor’s budget:



Project EXONE: contractor's budget	
Estimated cost of project	**€ 65,225**
Management reserve (10% of estimated cost)	€ 6,523
Total cost of project	**€ 71,748**
Mark-up% (% of total cost of project)	25%
Mark-up amount	€ 17,937
BUDGET (price to be paid by the client)	**€ 89,684**
Profit margin amount	€ 17,937
Profit margin % (% of budget/price)	20%

The price of the contractor’s work, 89,684 euros, includes a 20% profit margin for the contractor, which is equivalent to a 25% mark-up applied to the contractor’s total cost estimate of 71,748 euros (65,225 euros plus a 10% “Management reserve”).

- Project EXONE – Client’s budget:



Project EXONE: client's budget	
Internal costs	
Pre-3/1/17: Req'ts writing + Contractor selection	€ 4,500
3/1/17 through to 4/4/17	€ 44,650
Total Internal costs	**€ 49,150**
External & other costs	
Software development	€ 89,684
Content proofreading	€ 3,000
Product testing (beta & final)	€ 1,750
Server and software	€ 3,000
Multimedia asset licences	€ 2,000
Travel	€ 1,650
Total External & other costs	**€ 101,084**
TOTAL Cost	**€ 150,234**
Management reserve (10% of total cost)	€ 15,023
BUDGET	**€ 165,257**

In the client's budget, the cost of "Software development" is the price quoted by the contractor (89,684 euros) for its work on project EXONE. This budget also includes costs incurred by the client for writing the requirements specification and selecting a contractor, both of these tasks having been performed before the start date featured in the EXONE project plan.

Note that, as in the example above, costs are often broken down into "Internal costs" and "External (& other) costs".

Develop the human resource plan

This process, which uses the estimate of task resources as primary input, should identify the **skills and number of persons** required to successfully complete the corresponding project tasks. The human resource plan should also define the **organization of the project team**, in the form of a chart and/or matrix.

An example of organization chart is provided in chapter 10 ("The project team").

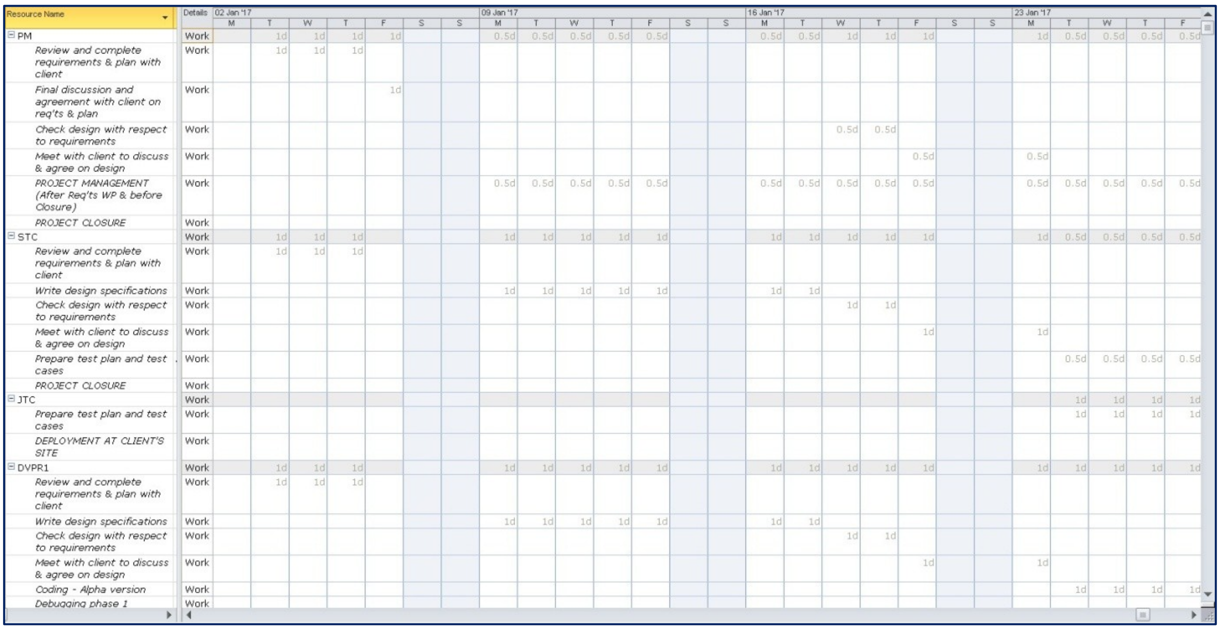
For each team member (not necessarily identified by name at this stage of the planning process), the following information should be documented:

- **role** (or function), for example consultant, developer, editor, tester;
- **position** in the organization, for example subproject manager;
- **responsibility**: the work to be performed;
- **competency**: the skills required to perform the work;
- **authority**: the degree of freedom (with set boundaries) for decision-making in areas such as resource assignment, methodology, quality control, communication.

The process should result in a "**staffing management plan**", indicating how staff will be acquired and featuring a **schedule for staff acquisition and release**, as well as requirements for training, team-building, etc. That schedule and those requirements obviously have a direct impact on the project's budget.

Some of the people required for the project may already be "on board", and some may not be available or not needed full-time (they may alternate between different projects).

The "**Resource usage**" feature of MS Project (or ProjectLibre) provides details of resources assigned to a project and the periods during which each person is busy on the project, as shown below for part of the contractor's work on project EXONE.

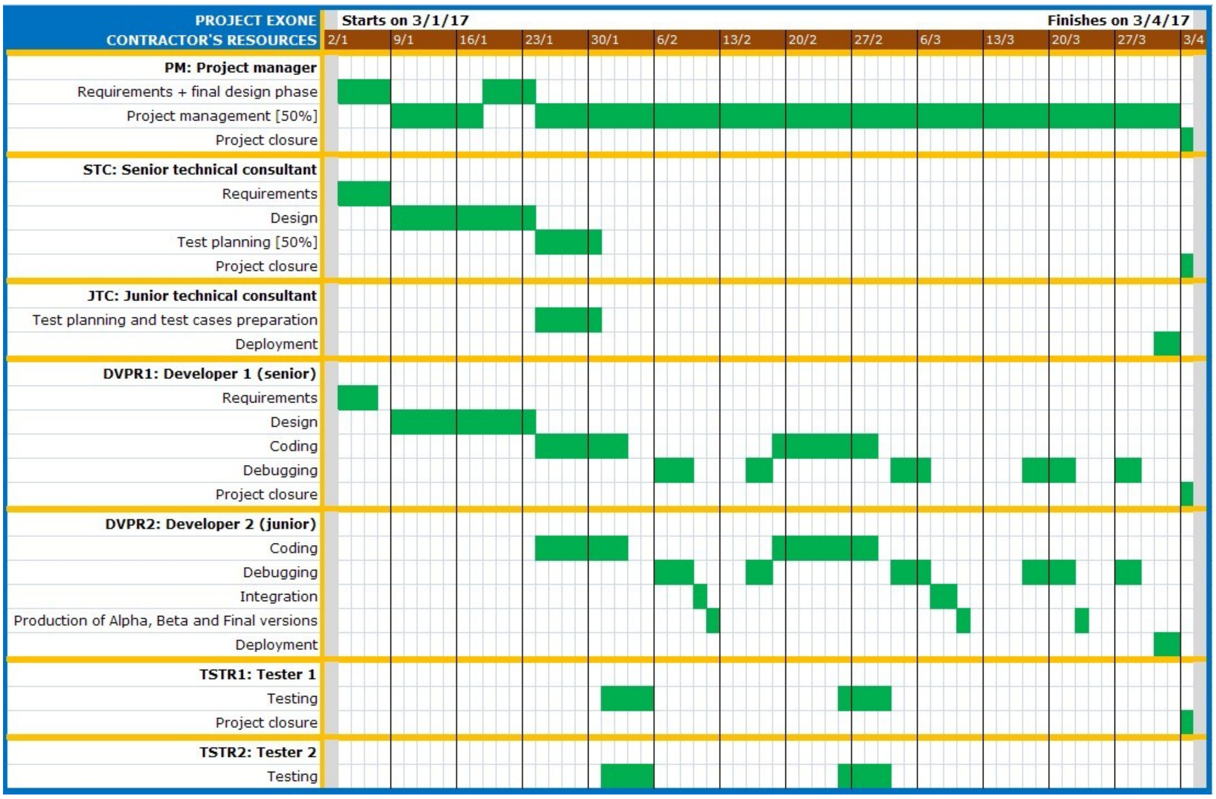


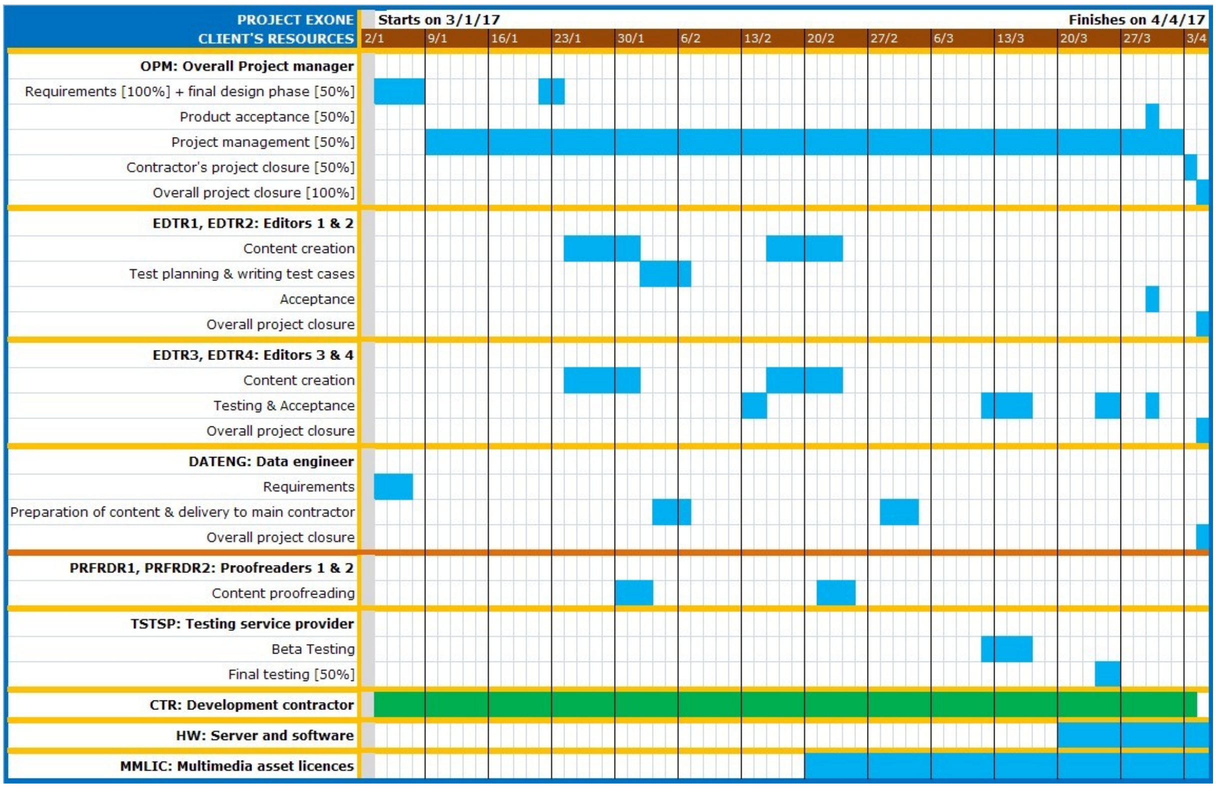
Resource Name	Details	02 Jan '17							09 Jan '17							16 Jan '17							23 Jan '17				
		M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F
PM	Work		1d	1d	1d	1d			0.5d	0.5d	0.5d	0.5d	0.5d			0.5d	0.5d	1d	1d	1d			1d	0.5d	0.5d	0.5d	0.5d
Review and complete requirements & plan with client	Work		1d	1d	1d																						
Final discussion and agreement with client on req'ts & plan	Work					1d																					
Check design with respect to requirements	Work																	0.5d	0.5d								
Meet with client to discuss & agree on design	Work																			0.5d			0.5d				
PROJECT MANAGEMENT (After Req'ts WP & before Closure)	Work								0.5d	0.5d	0.5d	0.5d	0.5d			0.5d	0.5d	0.5d	0.5d	0.5d			0.5d	0.5d	0.5d	0.5d	0.5d
PROJECT CLOSURE	Work																										
STC	Work		1d	1d	1d				1d	1d	1d	1d	1d			1d	1d	1d	1d	1d			1d	0.5d	0.5d	0.5d	0.5d
Review and complete requirements & plan with client	Work		1d	1d	1d																						
Write design specifications	Work								1d	1d	1d	1d	1d			1d	1d										
Check design with respect to requirements	Work																	1d	1d								
Meet with client to discuss & agree on design	Work																			1d			1d				
Prepare test plan and test cases	Work																							0.5d	0.5d	0.5d	0.5d
PROJECT CLOSURE	Work																										
JTC	Work																							1d	1d	1d	1d
Prepare test plan and test cases	Work																							1d	1d	1d	1d
DEPLOYMENT AT CLIENT'S SITE	Work																										
DVPR1	Work		1d	1d	1d				1d	1d	1d	1d	1d			1d	1d	1d	1d	1d			1d	1d	1d	1d	1d
Review and complete requirements & plan with client	Work		1d	1d	1d																						
Write design specifications	Work								1d	1d	1d	1d	1d			1d	1d										
Check design with respect to requirements	Work																	1d	1d								
Meet with client to discuss & agree on design	Work																			1d			1d				
Coding - Alpha version	Work																							1d	1d	1d	1d
Debugging phase 1	Work																										

Any resource overload will be flagged by the software in the Resource usage view, which is of course very useful and should lead to modifying resource assignment in order to avoid any such overload.

Since MS Project (or ProjectLibre) provides a view that may extend over several pages, as in the above example, it is also useful to create an **overview of resource usage** (if possible on a single page…), which can be derived from the project plan and easily prepared with a spreadsheet.

Here are two one-page **summary resource usage tables** for project EXONE, the first one concerning the contractor, the second one concerning the client.





In the above example, the client's resources include "external" people and other resources that are actually "procurements", which is the subject of the next section.

Note that activities related to human resources (staffing, training, etc.) take time, which needs to be taken into account in the overall project plan.

More information on human resources is provided in chapter 10 (“The project team”).

Plan procurements

This process consists in identifying project needs that can or must be met by **acquiring products or services from "vendors"** (or "sellers"), ie external suppliers, service providers and contractors, as opposed to needs that can or must be met by the in-house project team.
The process involves a "**make or buy**" analysis of tasks and resources.

The **procurement plan** documents the purchasing decisions, the methods to be used for **identifying vendors** (eg Request for Proposal (RFP), Request for Quotation (RFQ), Call for Tenders…) and the criteria to be used for **evaluating and selecting vendors**. It should also document the possible **risks** attached to each purchasing decision.

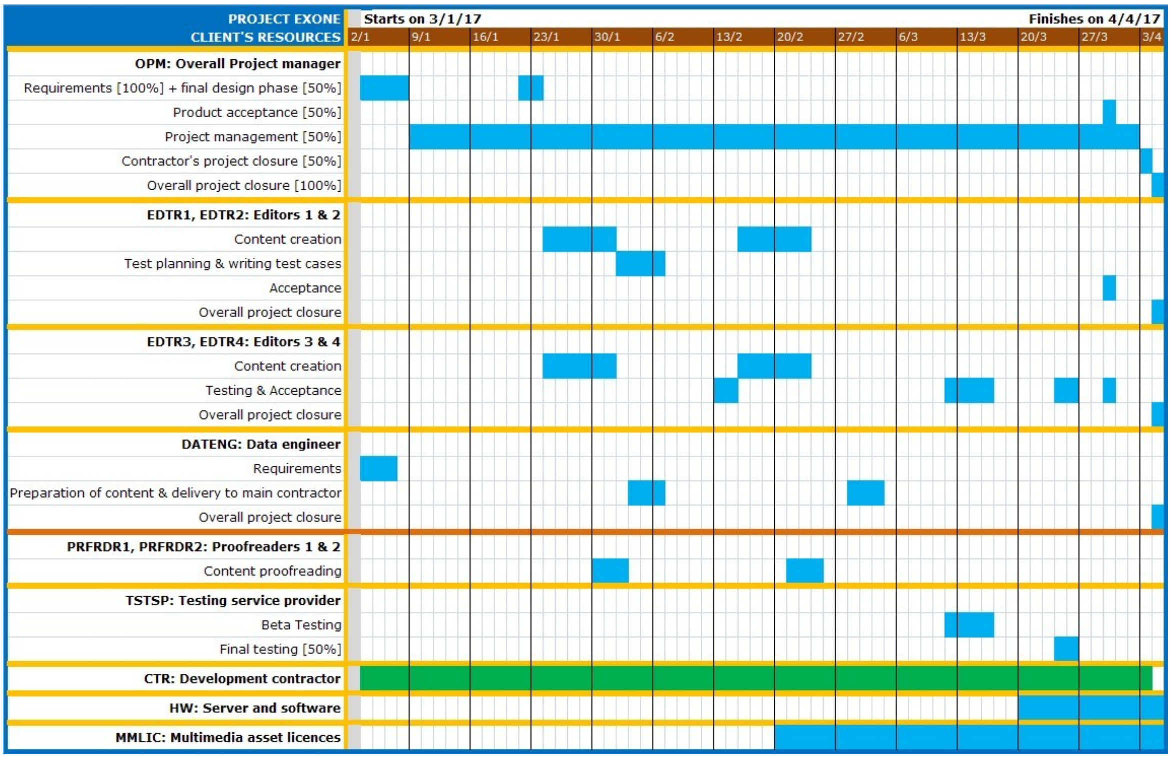
As mentioned in chapter 8 ("Requirements specification"), parts of the overall requirements specification document may be reused to prepare specific requirements documents for those of the vendors who need a formal "**Statement of Work (SOW)**".

The procurement plan should also indicate the **type(s) of contract and financial terms & conditions** to be used for the various categories of vendor.

Finally, the plan should include a **schedule** indicating at which stages of the project the various resources will be needed.

As mentioned for human resources, it is useful to provide an **overview of the procurement schedule** in tabular format.

The client's **summary resource usage table** for project EXONE below, already shown on the previous page, includes "external" people and other resources which are actually "procurements", namely the proofreaders, the testing service provider, the development contractor, the server & software, and the multimedia asset licences.



Note that activities related to procurement (preparation of SOWs, selection of contractors, drafting and negotiating contracts, etc.) take time, which needs to be taken into account in the overall project plan.

More information on procurement is provided in chapter 11 ("Contractors & contracts").

Plan quality

This process consists in identifying **quality requirements and standards** that will be applied to the project and to the resulting product, as well as documenting how **compliance** with such requirements and standards will be achieved and demonstrated.

The quality plan must of course take into account the **product acceptance criteria** specified in the requirements document and the **testing tasks** to be performed at various stages of the project. It may refer to the establishment at execution time of a detailed **test plan** and **test cases**, and, if applicable, it should take into account the requirement for any **automatic tests** that need to be developed.

As mentioned at the end of chapter 8 ("Requirements specification"), specific testing phases may be required, such as an "**operational acceptance testing**" (OAT) phase and an "**operational health check**" (OHC) phase. If such phases are specified by the project owner, they need to be featured in the quality plan. Work related to quality assurance during a product's **warranty period** must also be featured in the plan.

The plan should also describe the processes to be put in place for **quality control and assurance of the project** itself (as opposed to the resulting product).
For example, a well-thought-out **workflow**, with appropriate **validation steps**, eg for content creation or for software design and coding, contributes not only to meeting quality requirements for the product but also to the productivity of the project team.

Quality "**metrics**" may be defined for certain requirements of the project, such as compliance with the approved schedule and budget. A metric is a numeric value that can be measured. A certain degree of tolerance may be attached to a metric, corresponding to a maximum-allowed deviation from a standard that has been set.

The **cost of quality** ("**COQ**") corresponds to the cost of resources needed to meet quality requirements. Quality planning should therefore be performed in close relationship and interaction with the planning processes concerning tasks (some of which may be directly related to quality control/assurance, such as testing), resources, durations and costs.

For example, "pair programming", one of the recommended practices of eXtreme Programming ("XP") has a positive impact on the quality of coding, but a negative impact on resource requirements and therefore on cost (at least in the short term).
(See chapter 12 for more details on "XP".)
Likewise, proofreading is required to ensure a good quality of texts, but it has a price.

Poor quality also has a cost, which may actually be higher than the cost of quality (for example, **bugs can "kill"** a product or even people, or at least have a very negative impact on a company's image). However, a "**cost vs benefit**" analysis may lead to the conclusion that a given level of quality is sufficient.

> *A few months before the EHM was due to be released, a sample of its text content, including several hundred articles, and the same sample extracted from Microsoft Encarta were submitted to a team of proofreaders. The result showed that the average number of typos and other errors was almost the same for both samples, so the EHM was no worse than Encarta in this respect. However, the quality objective that had been set for the EHM content was not achieved, so the decision was made to have all of the EHM's articles proofread, which delayed the completion of the project by a few months and added a substantial amount to the editorial cost.*
> *A second round of proofreading would probably have been useful (given the total number of articles, over 40,000), but it was decided that the additional cost and delay would not be justified by the marginal benefit in terms of quality of content.*

More information on quality is provided in chapter 16 ("Testing").

Plan communications

This process involves determining what the **information needs** of the project stakeholders are, and **how, when** and **by whom** such information will be provided.

Information should be given in the **right format**, at the **right time**, with the **right impact**, to those people who "**need to know**", including, as appropriate, all or selected members of the project team. **Adequate and effective communication** requires time and money, which needs to be taken into account in the overall project plan.

The **communications plan** should provide the following information:

- types of information to be communicated,
- language(s) to be used,
- a glossary of terms and abbreviations specific to the project (to avoid ambiguity and misinterpretation),
- time-frame and frequency of communications,
- communication methods (meetings, memos, standard forms, e-mail, intranet…),
- persons with communication responsibility,
- escalation paths and procedures.

More information on communications is provided in subsequent chapters of this guide.

Plan risk management, identify and analyze risks

A risk is "**the possibility of something bad happening at some time in the future**", according to the Oxford English dictionary. Furthermore, "anything that can go wrong will go wrong", according to Murphy's law (see en.wikipedia.org/wiki/Murphy%27s_law).

Risks are inherent in any project. Obviously, **risks cannot be planned, but how to manage them can**.

Planning risk management is the process of defining **how to deal with project risks**. It also ensures that sufficient time and resources are allocated to risk management.

The **risk management plan** is closely related to other parts of the project plan such as those dealing with tasks, resources, procurement, budget and schedule.
A risk management plan should cover the following areas.

- **Methodology**: approaches, tools and data sources that may be used to perform risk management.
- **Roles and responsibilities** of team members who need to be involved in risk management.
- **Budgeting** for the costs involved in risk management.
- **Timing**: defines when the risk management processes will be applied in the project's life cycle.
- **Risk categories**: a structured list or table or diagram showing the various possible areas of risk.
- **Risk impact levels** should be defined, for example: low, moderate, high, very high or unacceptable.

The **impact of risks** relates to the effect they might have on the project objectives: scope, time (schedule), cost (budget), quality, etc.

The risk management plan should also include a **list of identified risks** (or "**risk register**"), preferably sorted by category, with attributes such as **estimated probability** (ie the likelihood of the risk materializing) and **impact level**, as well as actions already undertaken or to be undertaken for each risk, and of course an **evaluation of the cost of minimizing (mitigating) or eliminating risks**.

It is generally impossible in the planning stage to identify all of the risks that might impact the project, but the risk management processes that have been defined should **ensure that risks that appear in the course of the project will be dealt with appropriately**.

Defining risk management processes and identifying risks and associated costs involve **expert judgment**, the analysis **of historical data** (from previous projects), if available, as well as **teamwork** for reviewing the various components of the project management plan and tasks to be performed. **Brainstorming** workshops may be useful for this purpose.

The **cost** associated with a risk attached to a given item (task, work package, resource, supply, service, situation…) may be a **lump sum or a percentage** of the item's initial cost estimate.

Taking into account the **probability of risk materialization** is a tricky matter. For example, the possibility that a contractor used for the execution of a major part of a project might go out of business before completing its work on the project may be viewed as a low-probability risk, but its impact, should the risk materialize, might be very high. In such a case, as a PM, you should either neglect the risk (because of its low probability) or factor it into the plan with a cost corresponding to its full estimated impact level, regardless of its probability of occurrence.

Estimating the probability and impact levels of risks enables you to **prioritize risks** and thus to focus on those that are **most likely** to materialize and are the **most significant**.

Actions taken to **minimize** (mitigate) or even **eliminate** a risk may result in the **addition of tasks** (eg prototyping or testing iterations) **or resources** (eg editors, developers, testers) to the project, which will automatically result in **additional costs**.
It is good practice to make such additional costs **explicit** and to clearly indicate that they correspond to the estimation of risks.

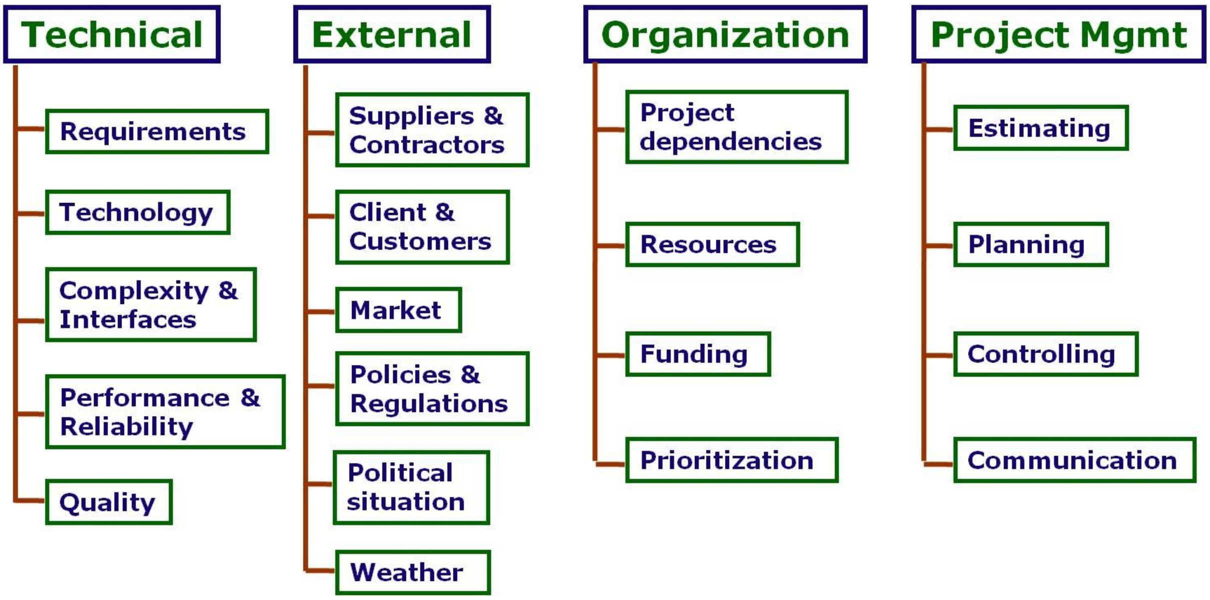
The elimination of a risk may require **changing the project's scope**, for example giving up some of the product's functions and/or features.

Instead of or in addition to work-package-level or task-level risk evaluation, a lump sum or percentage may be added to the project's total cost estimate, as a global "**provision for risk**" (sometimes also called "**management reserve**").

As a PM on the project owner's side, you need to make sure that the **requirements specification** mentions risks that have already been identified.
As a PM on a project implementation entity's side, you must take into account and evaluate not only those risks that are documented in the requirements specification, but also any additional risks identified as a result of **in-depth analysis** of the requirements, as well as those identified during the project planning process.

In the same way as the subdivision of work is represented by a WBS diagram, risk categories may be summarized in a "**risk breakdown structure (RBS)"**, which is a helpful **checklist** for the process of **risk identification**.

Below is an example of risk categories presented in the form of an RBS derived from the RBS documented in *A Guide to the Project Management Body of Knowledge ("PMBOK® Guide")* published by the Project Management Institute (pmi.org).



Examples of risk

A few **examples of risk** in some of the above categories are given below.

Requirements

Achieving website accessibility objectives may not be feasible due to technical constraints such as the use of JavaScript. If full compliance with accessibility standards is a "must have" feature but cannot be guaranteed by the developer, there is a risk that the product will be excluded from certain markets.
The risk may be eliminated by selecting a developer who can provide the necessary guarantee and is prepared, by contract, to accept to pay damages should the requirement not be met.
The risk may be mitigated by modifying the requirements so as to limit accessibility to a subset of the website's functions or to a subset of accessibility features.

> *The online version of the EHM was required to comply with W3C/WAI accessibility requirements for websites. The developer informed Hachette of the risk that full accessibility might not be provided because there might not be enough time and budget to evaluate and implement adequate technical approaches to accessibility for some of the EHM's features. In order to limit the risk, the accessibility requirement was dropped for the major issue at stake, namely the "Advanced search" function, which was typically of interest to only a small percentage of users.*

Technology

Advanced technology to be used in the implementation of a project may be an area of risk if it is not absolutely stable and/or fully mastered. If the project cannot be completed without such technology, the cost of the associated risk needs to be factored into the budget by estimating the additional time and/or resources required to fully develop and debug the technology, and/or required for learning how to incorporate it into the product being built.

Complexity and interfaces

The complexity of a project is obviously a key factor of risk. For example, a simple, straightforward static website development project is generally less risky than the development of a full-blown new information system with transactional applications intended to function in an environment with a complex network architecture.
Minimizing such a risk requires choosing the right development contractor, ie a company that has a proven track record of successful implementation of complex systems.

This risk category also includes **technical dependencies**, relating for example to the operating systems with which a software application is expected to be compatible. If a new version of an operating system (OS) is due to be released before the application launch deadline, there is a risk of incompatibility. In order to minimize such a risk, the developer should test the application with a beta version of the OS concerned and make any necessary code changes to ensure full compatibility.

> *The first version of the Hachette Multimedia Dictionary (DHM, a predecessor of the EHM) was initially released on the same day of August 1995 as Windows 95. Unfortunately, the DHM turned out to be incompatible with Windows 95!*
> *Fortunately, the problem was easy to fix, but the cost of the initial incompatibility was fairly high: a version "1.1" of the DHM had to be produced, a new batch of CD-ROMs had to be manufactured, and a product replacement process had to be put in place in order to deal with customer complaints.*
> *The developer was supposed to have tested the DHM with a beta version of Windows 95, but the work had not been done properly nor thoroughly checked.*

Suppliers and contractors

A given contractor may be working on several projects at the same time and could be tempted to reassign resources on a temporary basis to another client's project, because of an emergency situation. In such a situation, the risk incurred by the project at stake is a schedule slippage and a possible impact on quality due to lack of focus on the part of the developers.
Such a risk may be compensated by a provision in the contract for penalties to be paid by the developer in case the agreed deadline for product delivery and product quality requirements are not met.

> *Here is another example of risk and associated cost: based on previous experience working with a software development contractor, Hachette's top management wanted to reduce the risk represented by total reliance on an external development company for its EHM project.*
> *So management authorized the hiring of a software development engineer who would work in close cooperation with the development company in order to monitor project activities performed by the contractor, to participate in the development effort, and to facilitate the transfer of technology and knowhow, should development need to be internalized at some point in the future.*
> *It turned out, after completion of version 2 of the EHM, that the contractor decided to stop its multimedia software development activity, so production of version 3 was done internally, with a development team that included the above-mentioned engineer and three engineers who had worked for the contractor and who were hired by Hachette.*

Another example of a risk situation is that of a supplier supposed to manufacture and deliver a piece of new-technology hardware at a certain price by a certain date, but who may be unable to do so (at the specified price and/or by the specified deadline).
Again, such a risk may be compensated for by a provision in the contract for penalties to be paid by the supplier in case the agreed deadline for product delivery and product price specifications are not respected.
The risk may be minimized by choosing another supplier or delaying the product launch or increasing the estimated cost of its bill-of-materials.

Weather

Weather conditions may have a direct impact on certain projects, typically in the area of construction, but they generally have no influence (other than a possible influence on the morale of the project team…) on projects that are executed indoors, such as content creation and software development.
However, a contractor may be located in a part of the world where there is a risk of a natural disaster (earthquake, etc.). Depending on the estimated probability of a disaster and the importance of the contractor to the project, the risk may be accepted (possibly with a contingency reserve) or avoided by choosing a different contractor located in a naturally safer part of the world.

> *A company that did software development work for Hachette was initially based in California. Its offices and equipment suffered severe damage due to an earthquake in 1994, so it decided to relocate to Northern Florida (where it would only be exposed to hurricanes…).*

Project dependencies

Apart from obvious **intra-project dependencies** (eg a given task depending on the completion of another), the implementation of a project ("project A") may require deliverables resulting from another project ("project B"), which is an example of **inter-project dependence**.
Assume for example that some of the functions of a product to be developed require low-level software building blocks common to several products, such building blocks being designed and coded by software engineers in your company's R&D department.
As PM for project A, you have no control over project B. However, you need to identify and evaluate the risk of project B's deliverables being late and/or inadequate, which requires closely liaising with project B's PM as part of project A's planning process (as well as during its implementation).
Reducing the risk attached to such a situation may require a contingency plan ("plan B"!) featuring the development of an alternative solution to project B's deliverables (with obvious additional costs).

Resources

Hiring people can take longer than expected. A delay in the availability of the adequate people for the project can jeopardize its completion on schedule. Such a risk may be reduced by providing enough time for the hiring process and/or by making the jobs more attractive (in terms of responsibilities and/or compensation, with an impact on cost…).
The risk may be avoided by using a specialized agency that will generally be able to rapidly provide personnel on a temporary basis (but usually at a higher cost than direct hiring would involve).

Funding

Funding that is not fully guaranteed at the time a project is initiated is of course a major problem and risk. If a source of funds appears uncertain (or is believed to be unreliable), the cost of the related risk is at least equal to the amount of money supposed to come from that source. An obvious solution consists in finding an alternative source, which is however easier said than done!

A funding risk for a contractor may be a situation where its client's payment schedule provides for only a small percentage (say 10%) of the total fee to be paid upon signature of the project execution contract, while the contractor very well knows that the costs it will incur as soon as work is started or that it has already incurred in preparing the response to the RFP will not be totally covered by the initial payment.
In order to minimize or eliminate such a risk, the contractor will need to negotiate more favourable terms of payment with its client.

Should the client turn out to be inflexible, then the contractor may be forced to withdraw from the project, which may be safer than facing a cash-flow problem that may have dramatic consequences.

Estimating (resources, durations, costs, etc.) is **not an exact science**, so there is always a **risk of error** (which, in very rare cases, might even be a positive risk…). A negative risk may be translated into a **contingency reserve** to be added, for example, to the duration or cost of a given task.

> *As mentioned above (under "Estimating task resources"), a 15% margin of error was incorporated in the calculation of the duration and, consequently, resources required for the indexing of the EHM articles, following the measurement of a work sample.*

As a PM, your estimates may depend on input from other people, such as developers who are probably in a better position than you are to evaluate, for example, the time required to design and code a certain piece of software (unless you are not only a PM but also an experienced developer…). Most developers however tend to be optimistic, so a safe course of action may consist in having several developers provide you with distinct workload estimations from which you will derive your own estimation, using some technique such as the "Three-point estimate" (for what it is worth!) already described in this chapter (under "Estimating task durations").
Another more brutal method consists in adding a percentage, possibly based on experience with other projects, to the estimation provided by developers (or any other categories of people for that matter). For example, if the developers' estimate is 50 days, add 30% and assume that 65 days will be required!

Planning

This area is more general than just "estimating". It covers all facets of project planning, such as the scope definition, the work breakdown structure (WBS), the definition and sequencing of tasks in the various work packages (WPs), etc. Of course, **the longer the project** is expected to last, **the higher the risk** of planning errors being made.

Minimizing risks due to planning involves at least the following initial steps:

- provide (or obtain) a comprehensive and unambiguous requirements specification,
- create a comprehensive WBS,
- provide a comprehensive description of all WPs in terms of tasks to be performed,
- clearly identify all of the dependencies between tasks,
- identify the critical path(s) in the project schedule.

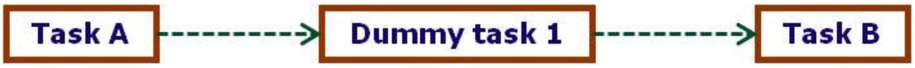
Any **uncertainty in the requirements** specification obviously **needs to be eliminated** before any serious planning can be undertaken.

There should be **no "holes"** in the WBS or in the description of WPs (the "100% rule" previously mentioned under "Creating the Work Breakdown Structure" needs to be applied).

Close attention must be paid to the **critical paths**, since they allow no margin of error should any of the corresponding tasks take longer than planned.

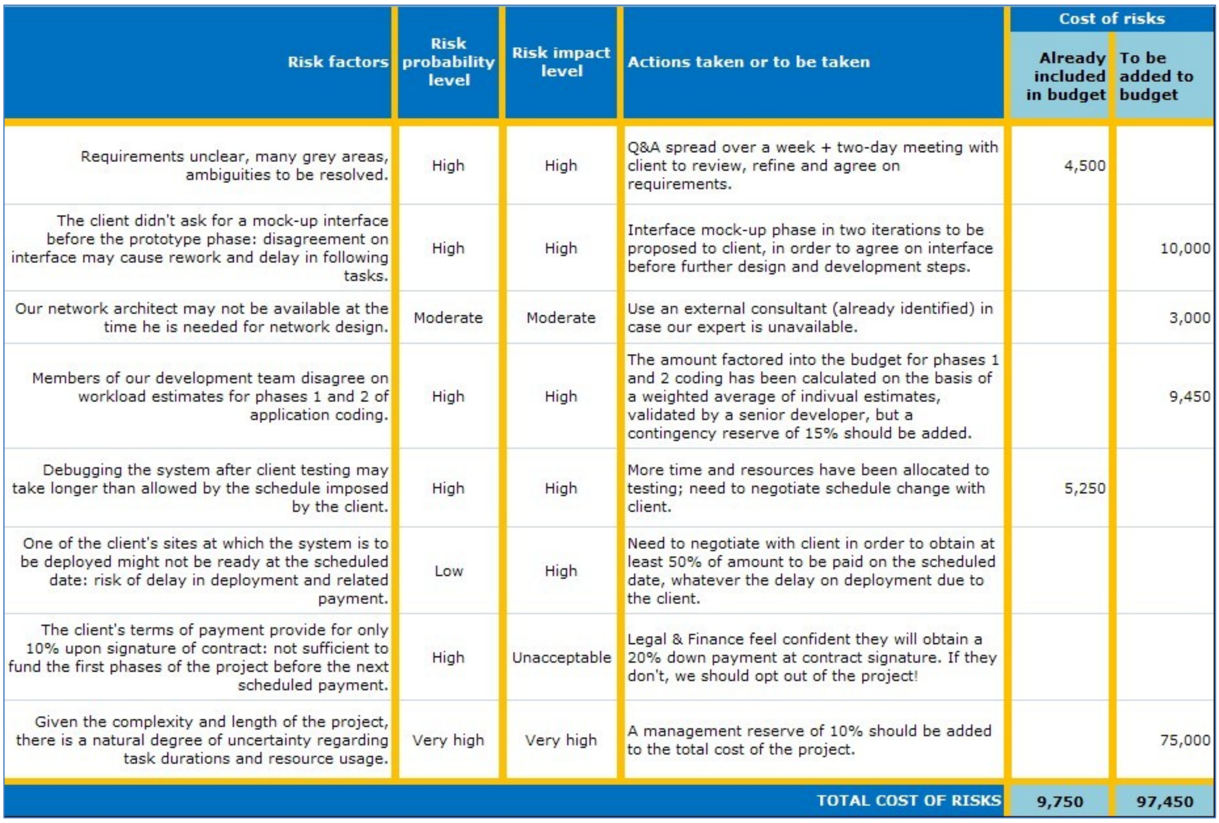
It may actually be useful to create "**cushions**" between dependent tasks. For example, you could add a number of well-identified **"dummy" tasks** between real tasks. The sole function of dummy tasks is to absorb any delay in the completion of preceding tasks, so dummy tasks should be given a duration but they do not require any resources.

Note that the connections between dummy tasks and real tasks in a network diagram are usually represented by a dotted line.



Example of risk evaluation

As an example, here is a summary presentation of a few risks identified for a hypothetical project on a main contractor's side, along with a description of the course of action to be taken for each risk and an evaluation of its impact on the project's costs, following an in-depth review of a client's requirements specification (in its initial state), and based on input from the Software Development team, Legal and Finance.



Risk factors	Risk probability level	Risk impact level	Actions taken or to be taken	Cost of risks	
				Already included in budget	To be added to budget
Requirements unclear, many grey areas, ambiguities to be resolved.	High	High	Q&A spread over a week + two-day meeting with client to review, refine and agree on requirements.	4,500	
The client didn't ask for a mock-up interface before the prototype phase: disagreement on interface may cause rework and delay in following tasks.	High	High	Interface mock-up phase in two iterations to be proposed to client, in order to agree on interface before further design and development steps.		10,000
Our network architect may not be available at the time he is needed for network design.	Moderate	Moderate	Use an external consultant (already identified) in case our expert is unavailable.		3,000
Members of our development team disagree on workload estimates for phases 1 and 2 of application coding.	High	High	The amount factored into the budget for phases 1 and 2 coding has been calculated on the basis of a weighted average of indivual estimates, validated by a senior developer, but a contingency reserve of 15% should be added.		9,450
Debugging the system after client testing may take longer than allowed by the schedule imposed by the client.	High	High	More time and resources have been allocated to testing; need to negotiate schedule change with client.	5,250	
One of the client's sites at which the system is to be deployed might not be ready at the scheduled date: risk of delay in deployment and related payment.	Low	High	Need to negotiate with client in order to obtain at least 50% of amount to be paid on the scheduled date, whatever the delay on deployment due to the client.		
The client's terms of payment provide for only 10% upon signature of contract: not sufficient to fund the first phases of the project before the next scheduled payment.	High	Unacceptable	Legal & Finance feel confident they will obtain a 20% down payment at contract signature. If they don't, we should opt out of the project!		
Given the complexity and length of the project, there is a natural degree of uncertainty regarding task durations and resource usage.	Very high	Very high	A management reserve of 10% should be added to the total cost of the project.		75,000
			TOTAL COST OF RISKS	9,750	97,450

> See the following articles for **more information on Risk Management:**

>> en.wikipedia.org/wiki/Risk_management

>> www.risk-doctor.com/pdf-files/hha0404.pdf

10) The project team

General remarks

The execution of most projects requires more than a project manager ("PM").
The expression "**Project team**" refers to the group of individuals working on a project.
The word "team" indicates that the individuals work in a collaborative manner with a **common goal: the successful completion of the project**.

For the duration of the project, the **PM** is the "**leader**" of the project team.

Project team members do not necessarily all work in a single location. They may be distributed over a number of locations, including homes for those individuals who "telecommute". Some locations may be "offshore".

By extension, the project team may include people working for **contractors** involved in the project. The PM is not expected to manage a contractor's human resources, but he may consider his primary contacts at the major contractor companies as part of the "**extended project team**" (so they will in particular receive information on a "need-to-know" basis and be involved in project team meetings whenever appropriate).

The **Human resources plan**, developed as part of the project planning process, includes a schedule for **staff acquisition and release**, corresponding to the needs identified for the various project tasks and to their estimated durations.

Team members may be "acquired", on a permanent or temporary basis, by various methods:

- an internal transfer within the company (from one department to another, from one project to another…);
- hiring from outside the company;
- employing independent (freelance) contractors (who may be fully integrated into the team);
- via a "staffing services" agency.

This chapter covers the **recruitment** and **integration** of team members, and the **organization** of the project team.

Recruitment

Recruiting qualified staff can be a **lengthy process**, all the more as suitable candidates may already have a job, so the hiring process should be undertaken as early as possible with respect to the date at which recruits are supposed to start working on the project. On the other hand, people should not be hired too soon, in order to avoid unnecessary costs.

The PM is generally in charge of the **recruitment process**, in close cooperation with the Human Resources department, which should provide guidance and any other form of help (including interviewing candidates and drafting employment contracts).

Subproject managers who are already part of the project team should be involved in the process of interviewing, evaluating and finally selecting candidates for their respective subprojects, but the PM is generally responsible for the final decision.

The **job descriptions** written as part of the human resources plan may need to be rewritten in more detail for the purpose of advertising available positions.
Note that **no confidential information** concerning the project should be included in job ads!

Job advertising can be achieved through different channels:

- internal job posting (at practically no cost, obviously);
- websites (at a very moderate cost);
- newspapers and magazines (at a much higher cost);
- networking (at a low cost);
- recruitment agency (at a very high cost);
- staffing services agency (at a relatively high cost).

In-house recruitment has a cost benefit; it also has the advantage of targeting people familiar with the company culture and, more importantly, who may be really motivated by the prospect of a job evolution (which does not necessarily imply a better salary, at least in the short term…).

Another very effective method for finding candidates is **networking**. The PM should use his own professional and social networks to spread the information concerning available jobs, and, for maximum leverage, ask his contacts (including the project team members already on board) to do the same via their respective networks.

In some cases, it may be necessary to use the services of a **recruitment agency** ("head hunter"). Such services are usually (very) expensive but they are generally efficient. It is important to provide the agency with a clear and detailed description of the background, skills and other qualities required for each of the positions to be filled, so as to ensure an adequate pre-qualification process and to avoid wasting time with unsuitable candidates.

Finally, **staffing services agencies** are usually able to provide specialized staff with a relatively short lead time. Such services are generally expensive, but they may be a good solution if other recruitment methods have failed, as well as in cases where, in the course of a project, there is an unexpected need for additional resources and there is no time for a standard (lengthy) recruitment process.

In preparation for the **interviews**, the PM should prepare a **summary description of the project**, to be used as part of a "sales pitch", whose purpose is to make the project and the jobs appear **attractive** to candidates.
The sales pitch should however be reasonable, ie it should not include promises that cannot be kept (such as a permanent employment contract at the end of the project…).
No confidential information must be disclosed during the interviews (as for job ads)!

Selecting **the right person** for each position should be based on a range of **criteria**, including the following:

- background/education,
- work experience,
- skills,
- achievements,
- references,
- passion (for products, etc.)
- team spirit,
- personality,
- availability,
- cost.

Note that a person with adequate skills but a somewhat difficult character may present problems working in a team…

It is desirable to have each candidate interviewed by several people, including the relevant subproject managers, other senior team members, and at least one HR person (skilled in detecting character flaws).

Integration

Integration of new recruits should be **as smooth as possible**, in order to make them feel comfortable in their new environment and at ease with the other team members, as well as to **facilitate** their **getting up to speed and becoming productive**.

New recruits should be introduced to the other team members on an individual basis and/or on the occasion of a team meeting. They should be welcomed as additional **valuable assets to the project**.

Prior to the arrival of new recruits, the PM should make sure that their **workstations** are **complete** (desk, computer, software, phone, Internet connection, intranet account, e-mail, etc.) and **in working order**.

New recruits must be given **adequate information** about the project, its context and purpose, the challenges it represents, its work breakdown structure (WBS), its organization, its schedule and, obviously, the tasks they will be involved in, as well as related objectives (duration, quality, etc.).

Ideally, the **information package** should be provided in written form and verbally explained by the PM or the appropriate subproject manager, who should make sure that the information is clearly understood.

Any **documentation** that new recruits may need to consult must be made accessible to them in a timely fashion (possibly in the form of direct links to documents on the intranet).

If any form of **training** is required for new recruits, for example on specific tools to be used, such training should be delivered as soon as possible.

It is also useful to give new hires the names of other team members who will provide **help**, if required. Of course, the people who have been designated to provide help should have been informed of this particular role they are expected to play, and they should have accepted it!

During the integration phase of new team members, the PM and/or the subproject managers should regularly check whether there are any **problems to be solved**, and they should take **appropriate action** when necessary.

Organization

The organization of the project team should be adapted to its size. It should be configured for **maximum efficiency**, avoiding unnecessary hierarchical levels.
It should be **optimized for getting the work done**, not for the beauty of the organization chart! It should also **facilitate communication and problem-solving**.

Responsibilities and levels of authority should be **well defined**. "Fuzziness" generally involves unnecessary overhead and leads to inefficiency.

The organization should reflect the project's **work breakdown structure**.
If subprojects are clearly identified, it may be useful or even necessary to appoint a manager for each of them. Ideally, a **subproject manager** should not only have good project and people management skills, but also be a specialist in the field which the subproject relates to, so that he can provide **expert guidance** to his team.

The PM will use the subproject managers as intermediaries for **relaying information and instructions** to subproject team members, as well as for **reporting information** on work progress, and for **escalating issues and problems** that cannot be handled at subproject level and therefore need to be resolved by the PM.

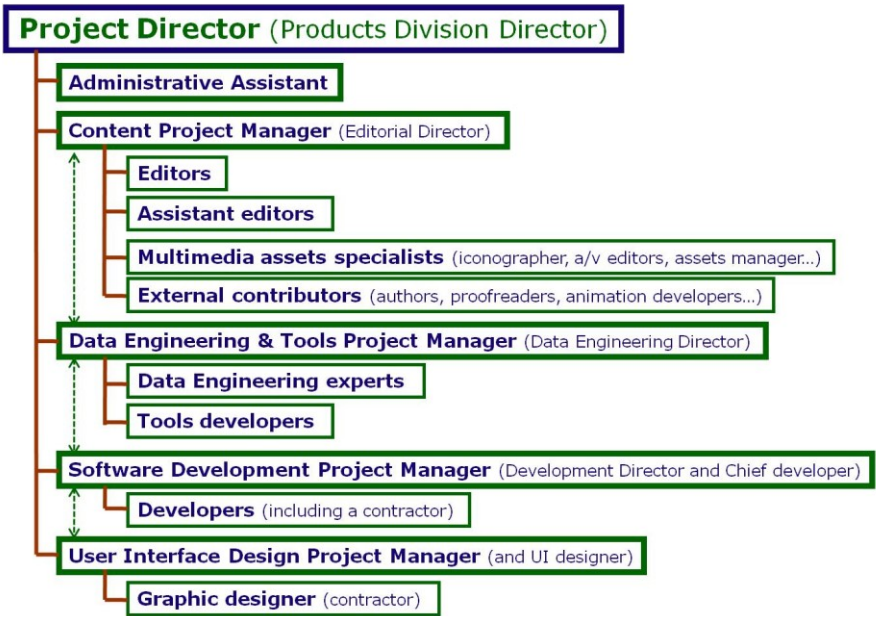
Ideally, the subproject managers should have **direct hierarchical authority** over their team members. Likewise, the PM should have direct hierarchical authority over the subproject managers.

The PM and the various subproject managers form the "**Project management team**".

In some organizations, relationships between the PM and subproject managers and/or between subproject managers and their team members are not hierarchical but **functional**, which can make the project more difficult to manage. In this case, **lines of authority** should nevertheless be clearly defined, accepted by all people involved and supported by the PM's manager.

> *At Hachette, I was in charge of a Products division. Among other assignments, I had full responsibility for the EHM project as Project Director. During the years when the EHM was produced in-house, I had several direct reports, including the Editorial director (who played the role of Content subproject manager), the director of Data engineering and tools (who played the role of Data engineering and tools subproject manager), the director of Software development (who played the role of Software development subproject manager), and a Project manager (who played the role of User Interface design subproject manager).*
> *The subproject managers had direct hierarchical authority over their respective team members.*
> *In addition to the above-mentioned subproject managers, I had an administrative assistant as direct report who took care of budget control, invoices and payments, as well as contracts (with support from the Finance and HR departments).*
>
> *The PM for another smaller-scale project at Hachette was one of my direct reports. Because of constraints imposed by the existing organization, the PM had no hierarchical authority over the other three project team members (an editor, a graphic designer and a website developer). The relationship between the PM and the project team members was purely functional. Due to the PM's character and lack of diplomacy, I had to get involved on many occasions to resolve personal conflicts and other issues that appeared in the course of the project.*

Here is an **example of a project team organization chart** (for the EHM project, at the time when software development was done in-house).


Project Director (Products Division Director)
Administrative Assistant
Content Project Manager (Editorial Director)
Editors
Assistant editors
Multimedia assets specialists (iconographer, a/v editors, assets manager...)
External contributors (authors, proofreaders, animation developers...)
Data Engineering & Tools Project Manager (Data Engineering Director)
Data Engineering experts
Tools developers
Software Development Project Manager (Development Director and Chief developer)
Developers (including a contractor)
User Interface Design Project Manager (and UI designer)
Graphic designer (contractor)

The dotted lines in the above chart represent the close cooperation that was required (and achieved) between the various subproject managers (and teams) for the success of the EHM project.

11) Contractors and contracts

General remarks

A project can seldom be executed with in-house resources only. Indeed, **specific competencies** required for certain tasks may be unavailable within the project team. Furthermore, it is sometimes more cost-effective to use **specialized contractors** to do a job within a limited time-frame than to hire specialists who may not be occupied full-time on the project and/or for whom there might not be a suitable position available at the end of the project.

Some companies may enforce a **policy** whereby hiring of personnel should always be consistent with the company's line of business. For example, most content publishers are not in the business of developing software or designing user interfaces or hosting websites, so if they undertake a project that requires software development, user interface design and website hosting, they will generally have the corresponding work done by contractors (or service providers).

In some cases, a contractor may belong to the same company as the project team. For example, a company's IT department may be awarded a contract for the execution of a project owned by another department.
In principle, such **intra-company contractors** should be put in competition with external contractors, unless the company's management wants to give preference to in-house resources, as a matter of policy or for other reasons.

> *Hachette Multimedia's participation, from June 2002 to November 2004, in a large European project ("CELEBRATE") subsidized by the European Commission, consisted in designing and producing a hundred or so "Learning Objects" for schools, in English and in French.*
> *Participation in this project was authorized by top management at Lagardere Active Broadband (the group that Hachette Multimedia was part of) on the condition that software development for the Learning Objects would be done by another company belonging to Lagardere Active Broadband, so that the bulk of the subsidy would benefit the group, as opposed to external contractors.*

Here are a few areas of activity for which contractors may be used: writing, proofreading, translation, art direction, graphic design, animation development, user interface design, software development, testing, website hosting, online payment system.

Statements of work and estimates

For most contractors, a formal **statement of work (SOW)** will need to be provided in order to describe the project and the work that is expected to be done. The SOW may reuse parts of the **overall requirements specification** document.
In the case of a **call for bids** or **call for tenders** or **invitation to tender (ITT)** or **request for proposal (RFP)** or **request for quotation (RFQ)** made available or sent to several potential contractors, the SOW should be the same for all candidates, in order to facilitate the comparison of the various **responses and estimates**.

The SOW should contain information about the context of the work to be done with respect to the overall project. If it must include confidential information, the companies or individuals to be consulted must be required to sign a **non-disclosure agreement (NDA)**, drafted by the Legal department or a legal adviser, before they receive the SOW.

The SOW should be **detailed, exhaustive and unambiguous**. The PM should require equally detailed, exhaustive and unambiguous proposals and price quotations from respondents. The PM may impose a **fixed format for responses**, so that they can be easily compared.

In particular, respondents should be required to specify for each task or work package documented in the SOW not only the price but also the **number and profile of persons** who will be **assigned to the work** and the time it is expected to take (this should give the PM an idea of the cost per hour or per day of the people to be involved in the work).

If the SOW includes multiple (possibly successive) **versions** of a product and/or **maintenance**, the respondents should be required to make the corresponding prices explicit.

Note that **prices** quoted by a contractor generally include a **profit margin**, which is an acceptable practice. The existence of a profit margin may provide **room for negotiation** (although obtaining a lower price than initially quoted may have negative consequences, as explained further on in this chapter).

Selection process

The process of selecting contractors obviously starts with making a list of companies or individuals to be contacted, except for a public RFP, in which case it is generally posted on a website for anyone interested to read and take action upon.

There are several methods for **identifying contractors** to include in the list of potential candidates:

- drawing on experience with contractors the company has dealt with on previous occasions;
- picking from a list of approved vendors;
- considering intra-company candidates;
- networking (asking around, using any available channel, such as the PM's networks and, by extension, his contacts' networks);
- searching the web;
- using the "Yellow Pages" (not the best method!);
- following recommendations.

Selecting candidates (individuals or companies) that other people **recommend** is generally a good approach. However, some people may want to do a favour to a friend or family member who is a freelance or works for some company, but who may not be the best candidate. The PM will need expert judgment and diplomacy to deal with such embarrassing recommendations…

As much information as possible should be obtained concerning potential candidates before consulting them, in order to avoid wasting time with companies or individuals that don't **meet the basic requirements** which should have been identified and listed in the "Procurement plan".

Responses from potential contractors are generally received by e-mail (or "plain old mail"). Some candidates may insist on presenting their response during a meeting with the PM. However, it is preferable to **analyze responses before any meetings with respondents**.

Indeed, the **first step in the response evaluation process**, as far as the PM is concerned, should consist in reading each of the proposals and checking whether it is complete and complies with requirements in terms of content and format. If it does not meet these conditions, the respondent should be contacted immediately and asked to send a new, complete and/or compliant version of his document.

When all proposals have been received (at the latest by the deadline set by the PM) and reviewed, the PM should **plan a meeting** with each of the respondents retained in the **short list**, ie those that have not been eliminated as a result of the first step of evaluation.

Each of these "first meetings" should preferably take place at the respondent's offices, so that the PM can get an idea of the kind of company he is dealing with.
Respondent's staff liable to work on the project should attend the meeting.

At some stage, the PM should **meet the "boss"**, ie the company's CEO or, in the case of a very large company, the head of the department where the work would be performed.

It is generally difficult to **avoid having to deal with a salesperson**. The rule to follow consists in **double-checking what the salesperson claims** by consulting people who would be directly involved in the work (a salesperson's job is to win contracts, not to do the work that the contracts entail!)

The PM should obtain and check **as much information as possible** concerning the candidate: previous projects and achievements, references, etc., and should enquire about other projects the candidate would have to execute in the same time frame as the project being discussed.

It is advisable to perform a **due diligence investigation** of a company before retaining it as a contractor, particularly if the work to be done by the contractor is critical to the project.
Such an investigation may be carried out by the PM's company's Finance department or by a Due Diligence consultant. Indeed, contracting with a company that may go out of business before the work has been completed may have dramatic consequences!

The way a potential contractor **reacts** to the SOW is another good test: the contractor's questions, requests for clarification, comments, etc. are to be taken into account in its evaluation. The absence of **relevant feedback or questions** from a potential contractor may indicate that the SOW has not been fully understood!

In the case of software development, candidates should be asked to indicate which **tools** they intend to use, and to justify their choice. They should also be asked to describe possible **technical constraints and dependencies** and to indicate how they would deal with related **risks** they may have identified.

> *One of the candidates for the development of the first version of the EHM proposed to use "PDF technology". The candidate was thanked for his time and promptly dismissed!*

A candidate's **geographical proximity** to the project team may be an important selection criterion, in particular when complex software development is involved, and even more so when substantial amounts of structured content need to be integrated with the software.
Such projects indeed require frequent meetings between the project owner and the development contractor. Although means of communication such as e-mail, wiki, videoconferencing, VoIP, etc. are very effective, distance may render collaborative work and project control difficult and costly.

> *The Hachette Multimedia Dictionary and Atlas were developed by a contractor based in Florida. At the time (1995-1996) there was no such thing as a wiki or Skype, so communication with the developer was done by fax, by AppleLink (an e-mail system for Apple and Apple partners, including publishers and developers) and by phone (in the afternoon and evening, Paris time, given the 6-hour time difference). On a number of occasions, it had been necessary to send several members of the project team to Florida to work with the developers, at a significant cost.*
> *Because of that experience, as well as for other reasons, the development contractor I chose for the first version of the EHM was a company based in Paris, France!*

Choosing a contractor involves **objective criteria** such as:

- the detailed proposal corresponding to the SOW,
- previous achievements,
- references,
- company size,
- financial situation,
- geographical location,
- the "price tag" and the detailed pricing breakdown.

Subjective criteria may also be applied, for example:

- the impression given by the company's personnel,
- the work environment in its offices,
- the degree of confidence displayed by its employees during discussions.

As regards the "price tag", many companies tend to practice "**low-bidding**" in order to win contracts. A contractor may decide to set a low price by **reducing its standard profit margin**, which might be dangerous for the contractor in the long term, but may not necessarily jeopardize the project.

A low price may also be obtained, sometimes involuntarily, as a result of **underestimating** the resources and time required to execute the contract, which generally has a negative impact on the project. Indeed, when confronted with reality, the contractor will usually try to reduce its project-related costs by sacrificing something, for example by assigning fewer people to the work, by limiting the duration of internal testing, etc.
As a rule, **the lowest price should not systematically determine the choice of a contractor!**

> *In 2005, Hachette needed to make a series of 15 or so CD-ROMs compatible with Windows XP, in order to be able to keep them on the market. An RFQ had been sent to two development contractors. One of the quotations was roughly ten times lower than the other! A quick analysis of the situation (involving a discussion with both developers) revealed that the lower bidder had completely misunderstood what the work consisted in!*

A useful tool for making the final choice of contractor is a **summary table comparing the candidates** with respect to the main criteria, whether objective or subjective, then weighing the "pros and cons" and finally **eliminating candidates** one after the other until there's only one left.

Depending on the extent and importance of the work to be contracted, the final choice, proposed by the PM in agreement with the appropriate subproject manager(s), may need to be approved by the PM's management.
Once the **choice** has been made and validated, the **PM takes responsibility** for it.

Contracts

A “**procurement contract**” is awarded to each of the contractors selected for the project. The contract may be a simple purchase order or a more complex and exhaustive document.

A contract is a **mutually binding legal agreement** that obligates the “seller” to provide the specified products or services in compliance with terms and conditions, and obligates the “buyer” to compensate the seller for said products or services.

Contracts are generally drafted by the Legal department or a legal adviser. The PM may however draft a contract himself using a previous one as a starting point and adapting it for the new project and contractor. The contract prepared by the PM will however **need to be reviewed and approved by Legal (and Finance)**.

Finalizing a contract often involves **negotiations before mutual agreement** is reached and the contract is signed. The PM generally participates in the negotiations, but does not necessarily lead them, in particular as regards financial and legal issues, which require experts in those fields.

A contract must be **exhaustive and detailed**. Here is a non-exhaustive list of topics usually covered in a contract (or its appendices):

- statement of work and deliverables,
- procurement schedule,
- pricing,
- payment schedule, terms and conditions,
- roles and responsibilities (of buyer and seller),
- subordinate subcontractor approval,
- transfer of rights and ownership,
- inspection and acceptance criteria,
- warranty, maintenance and support,
- change request handling,
- penalties,
- limitation of liability,
- insurance,
- confidentiality,
- termination and dispute resolution.

A contract should **provide for all contingencies** such as the contractor’s failure to complete execution of the work, to comply with the schedule, to meet the quality requirements, etc., and it should specify the corresponding **penalties and damages**.

For certain types of service provided by contractors, a "**Service Level Agreement**" (**SLA**) may be necessary in order to clearly specify, and quantify, the **levels of service required by the client from the contractor** as regards for example availability, support, response time, etc.
The SLA may also specify the **penalties** that will be applied if the contractor does not meet those levels of service.
The SLA may be part of the contract or an appendix.

In cases such as software development work, the contract should clearly specify whether **ownership** of the final application or system is to be transferred to the client or retained by the developer, and whether any **licence fees** will have to be paid by the client to the developer.
The contract should also specify whether **source code and documentation** is to be provided by the developer to the client.

The **compensation** for the service provided by the contractor may be a **flat fee** (fixed amount), possibly paid in several **instalments**, the first one usually paid upon signature of the contract, and the final one paid upon acceptance of the project deliverables in their final form, or at the end of the warranty period.

As mentioned in the chapter concerning the "P&L", a contractor's fee may be divided into a flat fee and a **proportional fee** (eg a percentage of gross or net revenues generated by the product), and, in some cases, a **licence fee**.

An additional fee is often charged by the contractor for **maintenance** beyond the warranty period (unless maintenance is covered by the fee corresponding to the development work).

Proportional fees are generally more complex to manage than flat fees, since they require execution of tracking, accounting, reporting and payment processes beyond the end of the project.
Contracts involving proportional fees may however have the advantage of reducing the initial cost of a project, sharing the risk of failure or lower-than-expected success of the resulting product, and motivating the contractor to do its best to ensure the quality of the product.

> *In 2004, it had become evident that both the CD/DVD-ROM and online versions of the EHM required a complete redesign and redevelopment to ensure compatibility with their environment and increase their attractiveness in a highly competitive market.*
> *However, Hachette Multimedia did not have the necessary budget to do that.*
>
> *The software development company that had been consulted during the feasibility study phase of this project was very excited at the prospect of such a challenging and interesting development project (it had previously developed an online version of both the EHM and the Hachette-Oxford dictionary).*
>
> *Talks with this company finally led to its accepting to be compensated for its work with a purely proportional fee (no flat fee), on the condition that a substantial advance payment would be made. Thanks to that financial arrangement, the project was approved by Hachette Multimedia's management, and was successfully completed.*

Reaching an agreement on a contract may take a long time, so it may be **tempting to start work before the contract is signed**. Doing this is **risky and certainly not recommended**, but it may be necessary if the schedule is tight and there is strong mutual trust between the parties involved.

Throughout the execution of a contract, **minutes** of important meetings should be taken and **decisions confirmed in writing**. These documents should be **filed for reference** in case of a dispute or litigation.

12) Software development life cycle models and methodologies

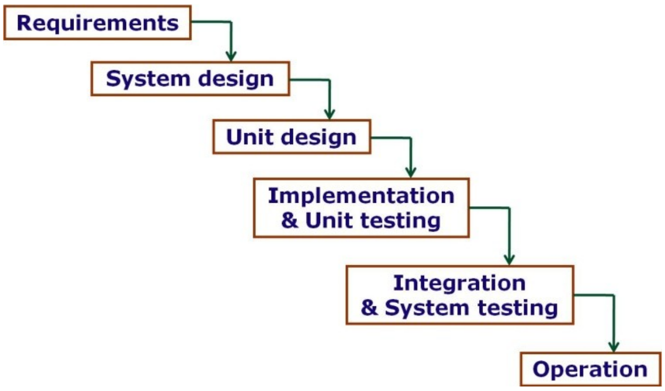
General remarks

As mentioned in chapter 3 ("The life cycle of a project"), there are many models that describe the **sequence of phases in a software development project**. Each particular model emphasizes certain features of a project's organization and dynamics.

The choice of a **model**, which has a definite **impact on project planning**, is dependent on factors and constraints specific to each project, for example the degree of flexibility allowed as regards the requirements specification.
An experienced PM will generally devise a specific model in order to take into account the characteristics of his particular project. Such a "home-made" model may be a variant of a "standard" model. A few standard models are described below.

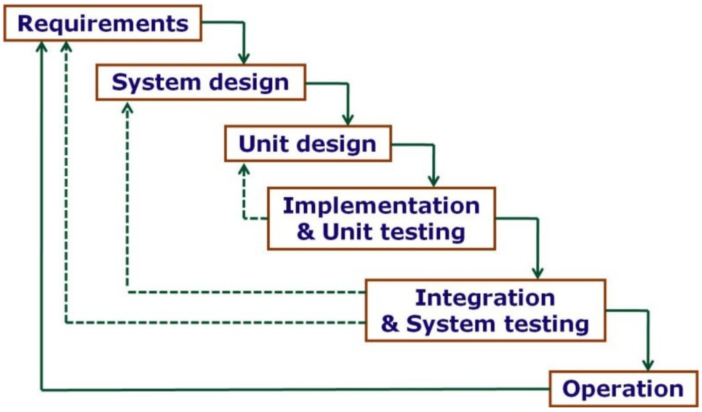
Waterfall model

The Waterfall model emphasizes the fact that each phase of a project must be completed, and its deliverables reviewed and validated, before the next phase can begin. For example, this model excludes going back and forth between product design and requirements, so there is **no flexibility**, which can be a major drawback for certain projects.



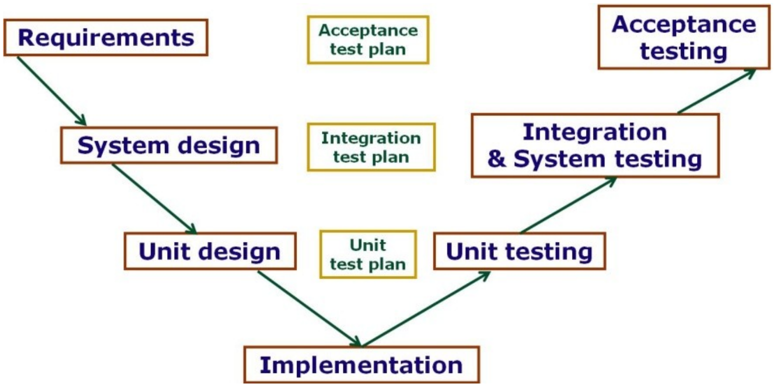
Incremental (or multi-waterfall) model

The Incremental model, which involves a repeated application of the Waterfall model, and therefore provides **more flexibility**, may be used for development projects that are executed in **multiple iterations**. Each iteration delivers a working version of the software (to be tested against specifications), which thus gradually evolves by increments into the final product, with limited risk of deviation from the requirements. One advantage of this model is that software is delivered for testing in stages, as opposed to being delivered "in bulk" at the end of the development cycle.



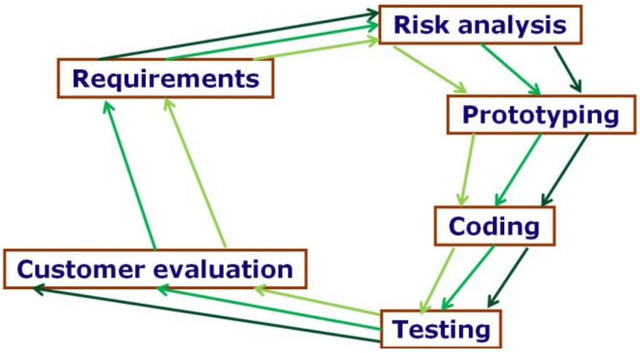
V (or V-shaped) model

The V model **emphasizes testing**: for each level of product specification, a test plan and test cases are prepared before implementation is started. Testing is performed at each level according to the test plan. The tests must first confirm that the "units" (or "modules") that have been developed comply with the unit design specifications. Compliance of the "system" resulting from the integration of all units, and content, if any, with the system design specifications must then be tested. Finally, compliance of the complete product with the requirements specification must be tested.
As with the Waterfall model, there is, in principle, no going back and forth between requirements and design, therefore **no flexibility**, which can be a major drawback for certain projects.
A **"double-V" (VV) model** may be applied to a project involving the development of a prototype then of a complete product; in this case the V model is applied successively to each development cycle.



Spiral model

The Spiral model, which is comparable to the incremental model, is an **iterative process that emphasizes risk analysis and resolution**. Risks (of a technical nature) are evaluated at an early stage and minimized as **multiple prototypes** are produced then tested and evaluated by the customer. The requirements specification may thus be adjusted to take into account what is technically feasible (or not). In this respect, the spiral model allows **more flexibility** than the three models described above.
However, the more "revolutions" in the spiral, the higher the cost of the project!



> See the following sites for **more information on Project life cycle models**:

>> codebetter.com/raymondlewallen/2005/07/13/software-development-life-cycle-models/

>> en.wikipedia.org/wiki/Waterfall_model

>> en.wikipedia.org/wiki/V_model

The above-mentioned models, which are useful for describing the successive phases of a project, feature various degrees of flexibility but are globally somewhat rigid. There are **other methodologies for software development projects** which allow **more flexibility and interactivity**, and which you, as a PM, should be aware of.
These methodologies need not be applied in a strict manner, but their general principles should be kept in mind. Some of the principles are actually featured to a certain extent in the models presented above, and many of these principles are just "**common sense**".

One particularly interesting methodology is "**Agile software development**", and "**eXtreme Programming**" (**XP**) is one of its applications. They are described hereafter.

Other methodologies (referenced by links at the end of this chapter) include "**Scrum**" (another application of "agile" principles) and "**CMMI**" (Capability Maturity Model Integration).

Agile software development

This methodology is based on **ten key principles**, listed and described below.

1. Actively involve users.
2. Empower the development team to make decisions.
3. Allow requirements to evolve but keep the project's timescale fixed.
4. Capture requirements at the highest level of description.
5. Develop small incremental releases, and iterate.
6. Make frequent delivery of product (to test…).
7. Complete a feature before moving on to the next.
8. Apply the "80/20 rule".
9. Integrate testing throughout the project's life cycle.
10. Rely on a collaborative approach between the project's stakeholders.

1. Actively involve users: this may seem obvious, but it is not always easy or even possible to involve users in all stages of a project, in particular during development. The more opportunities there are for users to test the product before its completion and to provide feedback, the better.

2. Empower the development team to make decisions: giving development team members a certain degree of decision-making authority, along with responsibility, contributes to motivation and efficiency, which is positive. A prerequisite is the involvement of the team at the early stages of the project, in particular the review and analysis of the requirements specification.
In most cases however, there will need to be a decision-making hierarchy within the development team so that if a consensus cannot be reached a decision can nevertheless be made.

3. Allow requirements to evolve but keep the project's timescale fixed: this means that flexibility is allowed with respect to the initial requirements specification. The specification may be adjusted as the project moves forward, in full agreement with the "client", in order to take into account changes that implementation has revealed to be necessary.
The timescale, however, should not be modified (unless the client agrees to such a change…), so that the final deliverable arrives on schedule (and within budget).

4. Capture requirements at the highest level of description: this means that requirements should not be specified in great detail (just enough to provide a general understanding of what they mean) at the beginning of a project; they will be refined at a later stage, after a number of iterations, each iteration resulting in deliverables that can be tested then possibly adjusted to match any requirements changes that have been decided and agreed upon.

5. Develop small incremental releases, and iterate: this is a very practical approach to development with iterations/increments ("analyze; develop; test"), one feature at a time (broadly speaking), thus limiting the risk of developing a product that does not meet requirements.

6. Make frequent delivery of product (to test…): this principle, which is included to a certain extent in the previous one, promotes a "perpetual beta" approach, so to speak, and is particularly applicable to website development, in which the first version of a site can be "open for business" within a few weeks after the start of the project, then can be adjusted and enhanced as required, integrating user/market feedback on a given version into the next version.

7. Complete a feature before moving on to the next: this principle emphasizes the fact that having developed a feature does not mean that it is complete; it needs to be fully tested and accepted before it can be considered "done"!
Of course, in some development projects, multiple features may be developed in parallel, in which case the principle applies to each "batch" of features.

8. Apply the "80/20 rule": this is an instance of Pareto's principle ("For many events, 80% of the effects come from 20% of the causes"), which applies to many areas of activity. In the area of software development, it may be interpreted as "80% of results (may) come from 20% of the development effort", which is often true. The general message is "focus your efforts on what is really important and provides maximum leverage!".

9. Integrate testing throughout the project's life cycle: this principle is consistent with at least two of the previous ones, namely "Develop small incremental releases, and iterate" and "Complete each feature before moving on to the next". The main objective here is product quality.

10. Rely on a collaborative approach between the project's stakeholders: this principle is actually a "must", given some of the above principles. Indeed, evolving requirements, iterative development, testing throughout the project cycle, etc. cannot be achieved without close cooperation between the development team and the other stakeholders (project owner, business analyst, user representatives...).

> See the following sites for **more information about Agile Software Development**:

>> agile-software-development.com

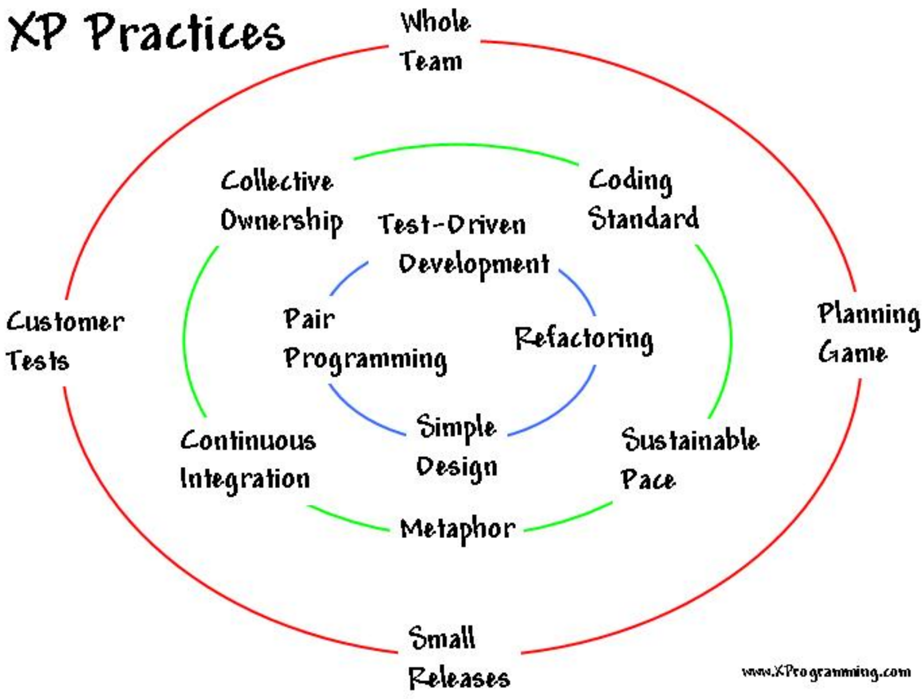
>> https://www.agilealliance.org/

eXtreme Programming (XP)

This methodology is a good example of application of Agile Software Development basic principles. It is summarized by the following quotation and diagram from a website referenced at the end of this chapter.

- ➢ XP improves a software project in four essential ways: communication, simplicity, feedback and courage.
- ➢ XP programmers communicate with their customers and fellow programmers.
- ➢ They keep their design simple and clean.
- ➢ They get feedback by testing their software starting on day one.
- ➢ They deliver the system to the customers as early as possible and implement changes as suggested.
- ➢ With this foundation, XP programmers are able to courageously respond to changing requirements and technology.

Note: in the last sentence above, the adverb "courageously" is used as a synonym of "fearlessly" or, by extension, "aggressively".



The keywords in the diagram represent the general principles of XP, which are briefly explained below.

Whole team: all contributors to the project form a single team, including at least one business and/or user representative.

Planning game: the emphasis is on steering the project rather than on exactly predicting what needs to be done and how long it will take. This involves two types of planning:

- Release planning: provides due dates for deliverables that are supposed to meet the requirements specification; the release plan may be revised by the "whole team", as necessary.

- Iteration planning: the direction given to the team is adjusted on a regular basis (for example every couple of weeks).

Simple design: software design is kept as simple as possible, for the initial release as well as for the following more complete releases, in order to always match the required functionality of the "system" being built (as opposed to wasting time on features that need not be included).

Metaphor: the development team should imagine and use a metaphor, or a set of metaphors, to describe in very simple and evocative terms how the software should work. The corresponding vocabulary should be agreed upon by the team.

Refactoring is a continuous process of design improvement that focuses on avoiding duplication and achieving full "cohesion" of the code developed, lowering the risk of problems that often arise when "coupling" (or interfacing) software units.

Continuous integration: the system must be kept "fully integrated" at all stages of development in order to maintain its cohesion. System builds are produced on a very frequent basis (several times a day if necessary).

Small releases: a working version of the software is delivered to the customer following each iteration; it is tested by "actual" users or user representatives, and may be accepted and even put into operation in order to gather as much real-life feedback as possible.

Customer tests are performed, as stated above, for each "small release" of the software. Furthermore, it is highly recommended to design and develop automatic acceptance tests that can be run over and over again to make sure there is no regression between releases. Automatic tests may complement manual tests, which are sometimes carried out too hastily.

Test-driven development: unit tests are systematic, with full coverage of the features that have been developed; as the system grows, so does the number of unit tests which need to be run successfully; feedback from the tests drives further development work.

Coding standard: code written by any member of the development team should comply with a general, unique standard, as if it were written by a single programmer. This approach ensures the cohesion of the system and makes code maintenance easier.

Pair programming: each software unit is developed by two programmers working together. The principle here is that "pairing" results in better code than would be produced by two programmers working singly, in about the same time frame, if not faster.

Collective (code) ownership: code produced by (pairs of) programmers should in principle be owned by all members of the development team, who are required to pay attention to code written by other members and to contribute to improving its quality.

Sustainable pace: developers should work with maximum productivity at a pace that can be sustained for long periods of time, thus minimizing the "burn-out" factor.
Note: this is easier said than done!

> See the following sites (from which the above quotation, diagram and explanations have been extracted) for **more information about eXtreme Programming**:

>> ronjeffries.com/xprog/articles/expdocumentationinxp/

>> extremeprogramming.org

> Visit the following sites to **learn about Scrum and CMMI methodologies**:

>> scrumalliance.org

>> cmmiinstitute.com

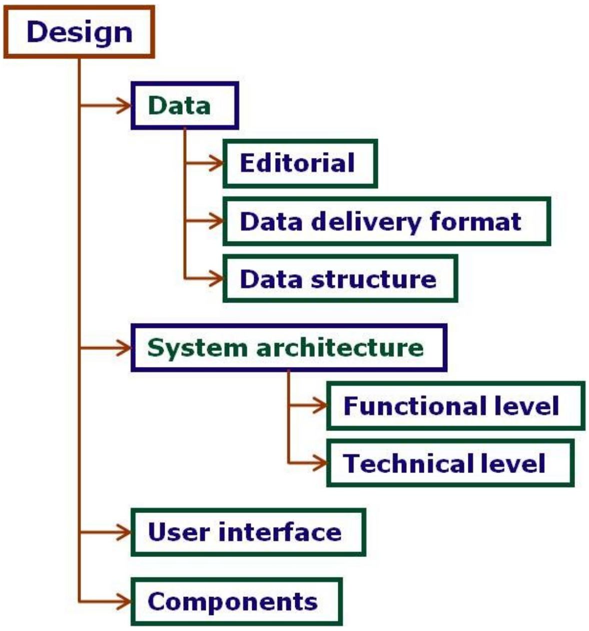
13) Design specifications

General remarks

This chapter addresses issues related to design specifications that are of interest to a PM, more precisely in the case of a project involving content creation and software development (in the general sense of the word, ie including website development).
It does not describe how to create content or design software, which are subjects generally covered in more specialized courses and manuals.

Design specifications are the "**blueprint**" for the work to be undertaken in order to **meet the needs expressed in a requirements specification**. As such, the design specifications describe in detail **how meeting the requirements will be achieved**.

Design specifications generally cover the following areas:



The purely functional description of the system architecture and the details of the user interface may be contained in a single document called "**functional specifications**".

The functional specifications and data delivery format specifications are of interest to all parties, ie not only the developers and the development subproject manager, but also the PM leading the overall project, as well as other subproject managers (and possibly team members) whose job involves a close relationship with developers.

The editorial specifications are generally of interest to content creators exclusively.

The data structure specifications, the system architecture specifications at detailed technical level, and the component-level technical design specifications are generally of interest to developers exclusively.

Specifications of interest to the project owner and the project implementation entities should be **reviewed and approved by the PM** and the project management team.

It is essential to ensure that design specifications are **exhaustive and precise**, are documented at an **adequate level of detail**, and **comply with the requirements** document.

Ambiguities and "grey areas" that might lead to misinterpretation should be clarified before sign-off.

It is strongly recommended to spend **as much time as necessary** on the design specifications, in order not only to **facilitate the subsequent product creation process**, but also to **avoid unpleasant surprises** with deliverables, and to **avoid potential conflicts** between the project owner and implementation entities such as content creators and software developers.

As a rule, a project implementation entity should never accept requirements that appear impossible to meet (given technical, schedule, budget or other constraints). Likewise, a project owner should never accept design specifications that do not comply with the requirements document.

Note that an **iterative process** is generally necessary in order to obtain a "perfect match" between design and requirements specifications. Indeed, specialists (editors, developers, etc.) involved in the design process may establish that some of the requirements cannot be met (due to technical or other reasons). If such a situation arises, solutions must be devised and agreed upon by all parties concerned, and the requirements document may need to be amended accordingly.

Once they have been agreed upon, the final design specifications of interest to both parties may be appended to the project execution contract.

Vocabulary/abbreviations

It is useful to provide a list of specific vocabulary (with precise definitions) and any abbreviations used in the specifications, in order to avoid any ambiguity or misinterpretation. That list may appear at the beginning of each of the design specifications documents or as an appendix, and/or may be stored for easy access in a central location (eg on an intranet or extranet site).

Editorial specifications

For a product with editorial content, for example an encyclopedia or an online newspaper or magazine, or an e-commerce website with a description of goods or services, as opposed to a software application without much editorial content such as a spreadsheet or a text processor or a web browser, it is necessary to write editorial specifications intended for authors, editors, etc.

Editorial specifications may cover for example (non-exhaustive list):

- the nomenclature (list of elements of content),
- content classification/hierarchy (sections, categories, topics…),
- guidelines relating to size, editorial style and other characteristics of texts,
- the style sheet for the presentation of texts,
- more generally, a style guide (or style manual),
- language(s) and localization constraints,
- links,
- metadata,
- DTD(s),
- style and format of multimedia assets,
- scenarios for animations.

Note that products generally assumed to be "without content", such as those mentioned above, may in fact include some content, for instance Help text, possibly illustrated, for which editorial specifications should be written.

Editorial specifications are a reference and a framework for people involved in **creating or editing content**. In particular, complying with the specifications will ensure a maximum degree of consistency among all elements of content, eg in terms of style.

Editorial specifications should match, but with more detail, the editorial section of the requirements document.

Editorial specifications are generally intended exclusively for people involved in content creation, not for software developers. They are usually written by the person in charge of the editorial subproject.

Data delivery format specifications

This document describes, at the lowest possible level of detail, how the **data** that will be delivered by the content provider to the software developer will be structured, tagged and formatted. It also describes the **metadata** associated with each data item or data set.
It may include a "**Document Type Definition (DTD)**" with a clear explanation of the meaning of each of the tags that will be used to mark up the data.
It is essential that both parties involved agree upon the data delivery format specifications before content creation and software development work is started, in order to avoid the cost of changes in the structure and tagging of content.

The data delivery format specifications may be written by the editorial subproject manager (or data engineering subproject manager, if any) or by the development subproject manager, or written as a collaborative effort between the two parties (content provider and software developer).

Note that the data delivery format specifications may actually be provided as part of or as an appendix to the requirements specification, as mentioned in chapter 8.

Data structure specifications

The data structure specifications are generally intended exclusively for software developers. They define the detailed structure of data for implementation purposes. These specifications should describe how data will be organized, stored, indexed, compressed, decompressed, encrypted, decrypted, retrieved, etc.

The data structure specifications are usually written by developers and/or the development subproject manager. In the case of complex data structures, an expert "database designer" may be involved.

Functional specifications

Functional specifications describe at a functional level the **architecture** of the software application to be developed, the detailed **behaviour** of its functional components, as well as the details of its **user interface**.
They describe how the **functions and features** expressed in the requirements specification will be implemented from a purely functional viewpoint.

The overall **architecture** of a software application (or "system") is comparable to the work breakdown structure of a project in that it represents the manner in which the system is divided into components (or units or modules) and structured. It should describe the manner in which components are related to one another and how they interact.

The system architecture can generally be described at the highest level with a **summary diagram**, and at lower levels with a series of more **detailed diagrams**.

Diagrams however are generally not sufficient to describe the precise behaviour of functional components. **Detailed narratives** must therefore be provided.

As regards the **user interface** of the software application (or system), a precise and exhaustive description of its elements must be documented, both **in graphical form and narrative form**.

The purpose of each of the **active graphical user interface** ("GUI") elements (menu items, buttons, checkboxes or tickboxes, data entry/input areas, etc.) must be explained in detail, as well as the corresponding function to be performed by the software.

Mock-ups of the user interface are obviously very helpful. In particular, they may be used as **guidelines for the graphic design work**, which also requires a precise description of the parts into which the GUI should be divided ("cut") and delivered for integration with the software by the developers.

The functional specifications are usually written by the development subproject manager (who may be working for a software development contractor). They should be reviewed and approved by the overall PM and the project management team.

In the case of complex systems, an expert "system architect" may be involved in writing system architecture specifications.

Note that the development team may require a more detailed and more technical description of the system architecture than that included in the functional specifications. The detailed system architecture may be a separate document or it may be included in the system component design specification.

Component specifications

This document (or set of documents) describes the functions that each of the software application's (or system's) components should perform and how such functions will be implemented.
It should also describe the relationships and interfaces between components.

The component design specifications are usually written by developers, for themselves or for other developers. If the development subproject manager is himself a developer, which is often the case, he may be involved in writing and reviewing the component specifications.

14) Content creation

General remarks

On many occasions in this guide I refer to "**content**" as a product component and to "**content creation**" as a project phase or subproject or work package. The objective of this short chapter is to make a few remarks on the subject, in particular concerning the **relationship between content creation and software development**.

The creation of content is often called "**content development**", but I prefer to use the expression "content creation" in order to clearly distinguish it from the development of the software component of a product, which can be described as the "**container**".

Note that when I use the word "development" without qualifying it, I mean "software development", not "content development" (likewise for "develop" and "developer").

The primary purpose of many software applications (including websites) is to make **content** available to end users. Projects of the kind this guide focuses on generally relate to products that feature some amount and form of content.

Content can be "**editorial content**", which globally refers to **text and multimedia assets** (photographs, drawings, maps, animations, videos, sound, etc.), such assets often being used to illustrate text, and sometimes being supplemented with text (eg captions for pictures or videos). It is usually created by people with **editorial skills**.

Content is sometimes simply called "**data**", for example commercial information to be fed into the database of a Customer Relationship Management ("CRM") system, to distinguish it from the "editorial" type of content as described above, although editorial content may also be considered as data.

"**Metadata**", including "metadescriptions", used for content indexing or for the description of illustrations or web pages, should also be considered as content, since it should be written by editorial-type people (as opposed to technical people).

There is also content that may be qualified as "**secondary**" or "**ancillary**", with respect to the "main content" (or "core content") of a product, but which is **nevertheless important**, for example: help text, credits, licencing and other legal information, privacy notice, sales terms & conditions, contact information, etc.
Some products feature no editorial content other than **Help** text (possibly with illustrations) and sometimes **templates** (eg sample presentations), directly available within the product or accessible via an associated website.

Finally, elements of the **user interface** of a product (eg menus, menu items, dialog boxes, text buttons, tooltips, warning and error messages, etc.) may also be considered as **content** insofar as the related text needs to be written by people with editorial skills.

Whatever the amount and form of content to be integrated into a product, the **content** needs to be **created and/or sourced**.

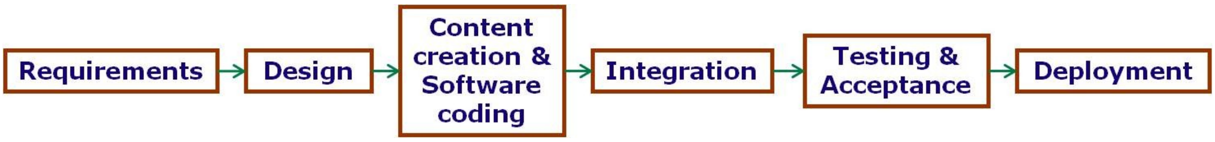
The entity that provides content is often referred to as a "**content provider**", which may be the **project owner**, for example a publisher, and/or a **third party** (possibly an "editorial packager" or an individual contributor) used as a **subcontractor** by the project owner to create and/or source content for a product.

The **content** of a product **is generally <u>not</u> provided by software developers**.
Even in the case of a product created by a software development/publishing company (eg Microsoft), content such as Help and templates is not created by software development engineers but by people with editorial skills (which developers generally do not possess…) who belong to an editorial department within the company or who work as or for a subcontractor.

Content providers and software developers need to work in **close cooperation** for the creation of any product featuring content as well as software.

Relationship between content creation and software development

As shown in the following diagram (which also appears in chapter 3, "The life cycle of a project", under "Project execution"), **content creation and coding (implementation) of the software part of a product** can generally be performed **in parallel** (by different people with different skills). These activities are usually featured as **distinct work packages and/or subprojects** in the project plan and organization.



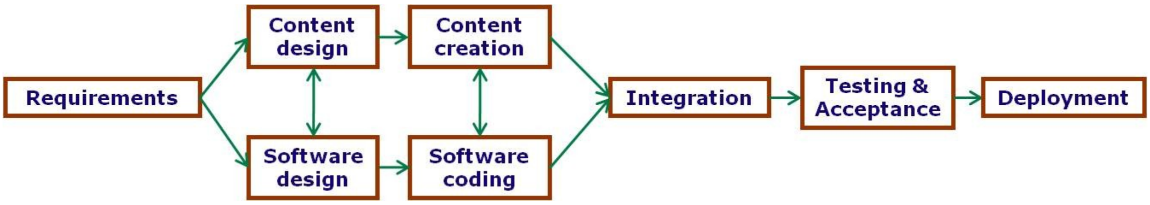
The "Integration" phase in the above diagram refers to the integration of content with the software, the result being a **complete product featuring both content and its container, the software**.

Note that content such as **Help and templates** cannot be created independently of the software since it is **directly dependent on the functions and features of the software**. The creation of such content can nevertheless be done partially in parallel with software development but it cannot be completed before the product is in its final stage of testing and debugging.

As mentioned in chapters 8 and 13, "content" is one of the many subjects to be addressed in a product's requirements and design specifications. Although some of the content design activities may be performed independently of the design of the software, there is a **strong dependence between the two areas of design**.

For example: software developers need to know enough about what a product's content will be in order to establish how it will be stored (possibly in a database), retrieved and displayed; developers also need a detailed description of how the content will be structured and tagged, as well as of categories of metadata that will be attached to content items (refer to the section "Data delivery format specifications" in chapter 13, "Design specifications").

The following diagram illustrates more precisely than the one above the relationships between **content design & creation** and **software design & coding (implementation)**.



Some content providers do not have any "**data engineering**" resources, so they lack the knowledge and skills to define and implement an adequate **data structure and format** for their content. In such a situation, developers may be required to do the necessary work in addition to actual software development. They may even be required to develop a set of **tools** to more or less automatically structure (or restructure) and format content as documented in the design specifications.

Some elements of content, for example **animations**, may require **software development work**. In such a case, editorial people will design and write the scenario of the animations, and the programming will be done by specialized developers, who may be different from those developing the product's main software.
Creating content such as animations is usually treated as a "**sub-subproject**" in a project's plan and organization.

Content delivery

At some stage in the development process, **sample content** will need to be integrated with the software, so that developers can check that content complies with the data delivery format that has been specified and that content-related software functions (store, index, search, retrieve, display…) work properly.

Sample content is often required by developers **very early in the development process**, sometimes even at the design stage, and certainly before the first "**Alpha**" version of a product is built, at which stage a **content subset** may be provided.

The **complete and final content** of a product should in theory be provided to the developer before the first "**Beta**" version of a product is built.

One **exception** to this rule is **Help** content, which should perfectly match all functions and features and user interface of a product as they appear in its final version. Help is therefore usually "frozen" in its final state just before the **final version** of the product is completed (which is often a challenge in terms of timing and project management…).

Content updates

The content of some products, in particular websites, needs to be enhanced and updated on a more or less frequent basis.

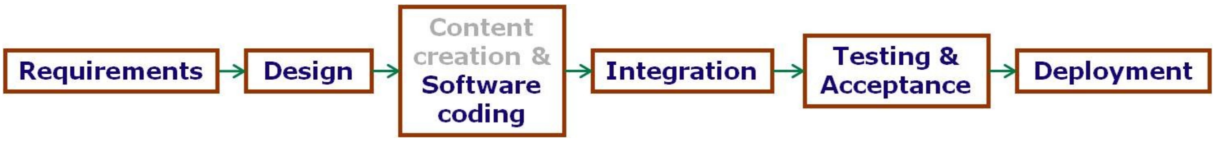
As mentioned in the "Product evolution" section of chapter 4 ("The life cycle of a product"), the initial project undertaken to create such products should include the design and development of a "**back office**" (or "back end"), which is a set of tools built to enable people in charge of creating, enhancing and updating a product's content (editors, etc.) to do so without requiring any intervention from technical specialists such as software developers.

Obviously, the back office needs to be **designed in close cooperation with its future users** so that the tools that are developed perfectly meet the editors' needs and make the task of content addition and updating simple and efficient.

15) Development and developers

General remarks

Projects involving the creation of software (including websites) require a type of work that is generally called "**development**" and covers the areas highlighted in the following diagram.



Software development is performed by people generically called "**developers**" (or sometimes "programmers", although in principle the term "programmer" has a more restrictive sense than "developer"). It is also referred to as "**software engineering**" and software developers as "**software engineers**" (or "software development engineers").

The word "developer(s)" is used in this guide to refer to an individual or a "development company" or the "development department" of a company or the "development team" that may be part of the overall project team in the case of in-house development.

The **technical skills** required of developers for a given project depend on the complexity of the software to be developed. Different competencies are required for the development of a static website, of a Flash animation, of a dynamic website involving a database and a sophisticated search engine, of a video game, of a software application involving complex algorithms, or of a sophisticated information system.
As mentioned in the chapters about human resources and contractors, the PM needs to **make sure that developers chosen for the project have the experience and skills required to meet the specific needs of the project**.

Internal vs external development – Proximity vs distance

If development is done internally (in-house), the PM can monitor it on a day-to-day basis and thus keep it under direct control, which is all the more important as development is often a major part of a project. The PM can ensure that developers are focused on the project, can identify problems or risks on the spot, and can react immediately.

Interaction between in-house developers and other project team members is easy and can happen as frequently as necessary. Regular or ad hoc meetings are easy to organize, practically at no expense.

Changes to requirements and/or design specifications, which may turn out to be necessary as the project moves forward, are easier to "negotiate" if development is internal, in particular if the PM has direct hierarchical authority over the development subproject manager.

If development cannot be done in-house (for cost or policy or other reasons), it has to be subcontracted to an external company.
In that case, considering the above-mentioned advantages of in-house development, geographical proximity should be a major criterion for the selection of a development contractor, particularly for complex development work.

Externalizing development does not necessarily mean using offshore companies, although offshore development is more and more frequent for obvious cost reasons, or because the experience and skills required for the project cannot be found elsewhere.

Communicating with developers

It is important to set up simple and effective means of communication with developers.

An **extranet site**, which may be a "**wiki**", is an excellent tool for **exchanging information** on the various issues that may arise in the course of development work, as well as for storing reference documents (requirements specification, design specifications, meeting reports, action items, list of contacts, glossary of specific terms and abbreviations, etc.).
Note that there are easy-to-use and relatively inexpensive tools for setting up a wiki, for example: www.editme.com/ and www.wikispaces.com/.

As regards the **delivery of content** to be integrated into the software (or website), small volumes of data may be uploaded to a wiki then downloaded by the developer. For large volumes, it is generally necessary to use an FTP server. In the case of "massive" volumes, it may be necessary for the developer to copy the data from the server where it resides, for example at a publisher's offices, to a large-capacity hard disk then transport it to the developer's facility and store it on the developer's server.
A development company may provide its own tools for uploading/downloading content; an example of such a company is IDM (IDM).

A **bug reporting and tracking system** should also be set up in due course (ie before the first tests are undertaken).
Mantis Bug Tracker is an excellent web-based tool for this purpose; it can be downloaded free of charge from www.mantisbt.org/.

That topic will be covered in more depth in the next chapter ("Testing").

The above-mentioned **tools** may be set up by the PM (or a person appointed by the PM) or by the development entity, which may have more experience in these matters, having probably done similar work for other projects.

Communicating with developers can of course also be done in meetings, with videoconferences, over the phone, using VoIP (eg Skype, WebEx), by e-mail and, if necessary, via instant messaging.

E-mail is an excellent means of exchanging information, of asking questions, of providing answers and, if the need arises, of giving instructions. However, as a rule, inundating developers (or any other person or contractor for that matter) with e-mails should be avoided!

The style used in e-mails is generally rather informal, which is an accepted practice, but this should not preclude **rigour and precision!** E-mails sent to developers should be **short and focused**, ie limited to what is essential (developers should spend most of their time developing, not extracting useful information from messages they receive…).

As far as possible, an e-mail should address a **single issue**, especially if it is intended for a developer (as opposed to the development subproject manager). Indeed, developers (like many other categories of people…), have a tendency not only to deal with one point at a time but also to retain a single point, usually the first one, among the multiple issues that may appear in a message. **Effective e-mails** are those that are extremely well targeted.
The above remarks also apply to messages sent via other channels, eg a wiki.

All important e-mails exchanged with developers (and with all other contractors for that matter) should be **archived** (and perhaps even printed), so that they may be subsequently used in case of a controversy or dispute.
Likewise, if an **instant-messaging** "chat" is organized with developers to discuss something of importance, all of the messages exchanged should be copied & pasted into a text file for future reference, should the need arise.

By the way, a **PM with a technical background** will feel more comfortable and be more effective in communicating with developers and/or a development subproject manager than a PM who lacks such experience. As a PM, you should at least be able to **understand developer jargon** (being fluent in C++ or Java is however generally not required!).

Monitoring and controlling development work

The PM should **give close attention to the progress of development work**, especially if it is a major part of his project. He should monitor and control development on an almost permanent basis.
The PM will naturally be involved in the initial stages of the development effort (eg reviewing and discussing functional design specifications), as well as in its final stages (testing and acceptance), but it is not easy for a PM to keep up-to-date on what is going on during the intermediate phases, when the "developers are developing".

Indeed, developers like to be left alone while they are writing code!

It is of course the development subproject manager's job to monitor and control the work of the development team, but the overall PM too should make sure, on a regular and frequent basis, that development is progressing smoothly in compliance with the agreed schedule, that no insurmountable problem has arisen and that no shortcut has been taken which might jeopardize the quality of the product to be delivered.

Paying regular "**courtesy visits**" to the developer is a good method of "gauging the temperature". A more formal approach consists in featuring **checkpoints** in the development plan, ie **milestone dates** at which the developer is expected to show the PM part(s) of the software (or website), even before its official versions are supposed to be ready for testing.

In this respect, in-house development has a clear advantage over external development, as mentioned at the beginning of this chapter.

As soon as **testing** begins, the evaluation of development work can be performed directly on the product being developed. Testing should be conducted rigorously and exhaustively, as detailed in the next chapter.

Successive versions of software

Software development is performed in stages and increments. The application (or system) is progressively created by developing **building blocks** (or components or units or modules) and assembling them. Some of the building blocks may be fundamental tools that make up the "lower layers" of the software, for example a database manager, a search engine or a display engine.

It is important to check that each building block is "**rock-solid**" and complies with the design specifications. This is done by the developers themselves and/or by the development subproject manager, who organizes testing, manages trouble-shooting and ensures quality control.
The overall PM is generally not involved in component testing. In some cases however, an early version of very important components, possibly featuring a basic temporary interface, may be made available to the project owner for testing and feedback.

> *For the EHM, a preliminary version of the search engine, using a representative data sample, was tested by Hachette in order to check that its principles of operation complied with specifications, and that its performance was satisfactory.*

The development schedule specified in the requirements (and possibly in the contract) should include **major milestones** corresponding to the **successive "official" versions** of the software. Those **deliverables** consist of predefined sets of functions, features and content, as specified in the requirements. They should be **thoroughly tested by the developers** before being delivered to the project owner for testing and feedback.

As an example, here is a typical list of successive milestone versions for a software application development project.

- Alpha: consistent subset of the application's user interface, functions & features and content (there may be several alpha versions after the first, each one integrating the correction of problems detected in the previous version).
- Beta: complete user interface, function & feature set and "final" content (there may be several beta versions).
- Release Candidate (RC): complete application integrating the correction of problems detected in the final beta version (there may be several RC versions).
- Gold Master (GM) or Final: complete application ready to be installed or duplicated for distribution on CD/DVD-ROM or any other medium, or, in the case of a website or web service, ready to be published online; the GM is validated and accepted by the overall PM following resolution of problems detected in the final RC.

A **prototype** may also be required by the project owner as a "**proof of concept**" before the "**real product**" is developed. Prototyping has the advantage of minimizing technical risks and testing the specifications. It may lead to a **revision of the requirements and design specifications** for the development of the "real" product.

Documentation and source code

A development contract generally requires the developer to deliver not only a software application (or system or website) but also documentation relating to the software as well as its source code.
In theory, those elements may be used by the client in a situation where maintenance or a substantial modification of the software is necessary but cannot, for some reason, be performed by the original developer.

The contract may however stipulate that the developer should retain **ownership** of the software, and therefore of its source code. In some cases, a developer may retain ownership of only part of the software, eg a proprietary search engine.
Software owned by the developer is generally used by the client under **licence**, for which there may be a fee (as mentioned in chapter 11, "Contractors and contracts").

Warranty and maintenance

Development contracts should include a **warranty** clause whereby the developer is obligated to **fix bugs at no additional expense** with the shortest possible delay during a determined period (the "**warranty period**").
The contract should also include provisions for the **maintenance** of the software beyond the warranty period. Maintenance may also be the subject of a separate contract.

There may be several categories of maintenance, eg "**corrective maintenance**" (bug fixing) and "**evolutive maintenance**" (eg to ensure the software's compatibility with its environment as the latter evolves: new operating systems or plug-ins, etc., or to produce new versions of the software, as required by the client).

A maintenance contract may specify the maximum response time required of the developer for given categories of problems (eg two hours for "critical" problems, one workday for "serious" problems and one workweek for "minor" problems).

Developers should obviously be compensated for the maintenance service they provide with a global flat fee, a flat fee per intervention or a fee per hour or per day.

16) Testing

General remarks

Any **product must be tested** before it is made available to the users it is intended for.

It is very likely that a product that has not been sufficiently tested will feature major defects. Such defects may "**kill**" a product (sometimes even people...) and convey a very negative image of the developer and/or of the project owner (publisher, etc.), even if the product is free of charge (eg an institutional website or a website with adverts).

Testing is therefore a **fundamental** activity of any project. The **PM**, as **guarantor of the product's quality**, must ensure that testing is well planned, well organized and performed in depth and exhaustively. Furthermore, the PM must **ensure that problems revealed by the tests are reported and solved** by whomever is concerned (content provider, developer, graphic designer...).

In principle, **developers should test the code they have written and debug it** before their software is made available to the project owner.
However, as a rule, **the PM cannot rely on developers to thoroughly test the result of their work**. Indeed, developers tend to expedite testing, even if it is organized and controlled by a Development subproject manager.
Testing by developers must therefore be supplemented with testing organized and controlled by the PM, which is the focus of this chapter. The perspective is that of an overall PM working for the project owner.

Test plan – Test cases

A **test plan** and "**test cases**" (or "test scripts") need to be prepared for each of the **major deliverables in the project execution process**.

The test plan should include a testing schedule and it should specify "who tests what". The test cases should describe in detail the elements to be tested, what the tests should consist of, as well as the expected results of each test.

Note that preparing a test plan and writing test cases are tasks that should be explicitly featured in the **overall project plan**, along with the actual testing tasks, as part of a "**Testing work package**" for each deliverable.

Obviously, **adequate resources** need to be assigned to testing, which is generally very time-consuming. The time and resources (mainly people) required for testing are **often underestimated**.
Note that trainees (interns) may be assigned to parts of the testing effort if the PM is certain they will do a good job (some guidance and control may however be necessary).

Internal testing vs external testing

The PM may subcontract testing to a **service provider**, in particular, in the case of software, to check that the product works properly on a broad variety of platforms not available in-house. A service provider may also be used for performance testing, load testing and for accessibility testing (which usually requires the involvement of visually impaired persons).

If an external testing service is to be used, the scope of the tests to be subcontracted should be extremely well defined, in particular with precise test cases. The service should be limited to testing that cannot be performed in-house, unless the PM believes that some degree of duplication may be useful and that the corresponding cost is justified.

Experience shows that external testing, though extremely useful and even absolutely necessary in many cases, is generally not sufficient, in particular as regards editorial testing and even functional testing of complex products. Successive versions of software must therefore also be tested **internally** (in-house), **not only by the developers but also by the project owner**.

Types of test

There are several **types of test**, in particular:

- editorial tests,
- content integration tests,
- user interface tests,
- functional tests,
- technical tests,
- performance tests,
- documentation tests,
- accessibility tests,
- free tests.

For all types of test, with the exception of free tests, **compliance** of the tested elements **with the requirements and functional specifications** needs to be checked. The specification documents, in total or in part, must therefore be made available to testers for reference as needed, unless the test scripts provide sufficient information.

In certain cases, **automated testing software** may be used to perform automatic tests of successive releases of a product. Such software must obviously be designed and developed with extreme rigour, and, of course, it needs to be thoroughly tested!

The expression "**black-box testing**" is sometimes used to refer to functional (or behavioural) testing, which does not require any knowledge of the software's internal structure and coding, as opposed to "**white-box testing**" (or structural testing, aka clear-box testing, glass-box testing, transparent testing), which requires internal knowledge of the software and therefore can be performed only by the developers themselves or technical persons specialized in such testing.

Editorial tests

Editorial tests concern the product's **content**: texts, images (and captions) and other multimedia assets, such as animations, for which compliance with the script must be checked. Editorial tests should also include testing links between elements of content, in particular those that have been established manually.

Editorial testing also applies to user-interface text elements, as well as user manuals, Help, tutorials, "Read me" documents, etc. These items will be addressed in more detail further on in this chapter.

Editorial testers must be able to **read and write properly** (in at least one of the languages in which the content is written), so they should preferably be professional proofreaders or editors.
Note that **developers and technical people in general are rarely good writers** (they tend to make spelling and grammatical errors) **and are** therefore usually **poor proofreaders**. The same is true of testing service providers, unless of course they are specialized in editorial testing.

Editorial tests may be extremely tedious for products with a substantial volume of content, but they must be performed in a rigorous and exhaustive fashion.

For certain products, **content may be tested independently of the "container"**: for example proofreading of texts and testing of animations may be done before the content is integrated with the software.

In the case of a website, it is generally possible to change content after the site has been released, so editorial bugs that have not been corrected before the end of the product creation process may nevertheless be fixed at a later stage, in principle without too much effort.
Post-release editorial bug fixing is obviously made easier with **"back-office" tools** that enable editors to change content without any intervention on the part of developers.

Content integration tests

As of the first alpha version of the software, it is advisable to check that **content** has been **correctly integrated** and that its **appearance and layout** comply with the specifications, in particular as regards the **style sheets** and, more generally, the **style guide** (or style manual).
Although content integration testing can be performed as early as the first alpha version of the software, it cannot be done exhaustively until the **first beta version** is delivered by the developer, since alpha versions do not include all of the product's content.

A typical problem to look out for is the **degradation of special characters** (accented characters such as "É" or ligatures such as "œ"), and even apostrophes. Such degradation on a website may however be due to a bad choice of character encoding…
Note that special characters often pose a problem in search functions, so searching for words featuring such characters should also be thoroughly tested (this is particularly important for products featuring a full hypertext capability, where a click (or double-click) on a word triggers a search for that word).

Another potential problem to be submitted to testing is the **degradation of content due to digitization, reformatting, compression** or any other processes that may not have been performed properly. Content integration tests should therefore also cover the **rendering of multimedia assets**: sharpness of images, fluidity of animations and videos, quality of sound.

Content integration tests may be very tedious but they should be carried out systematically and with extreme care.

User interface tests

These tests apply to all of the elements that make up the user interface of a product: its overall architecture (from the user's viewpoint), navigation tools, ergonomic features, menus, buttons, checkboxes (tickboxes), tooltips (or "bubble help"), error and warning messages, graphic elements, etc.

All of the text elements of the user interface must be proofread for each language by testers capable of detecting grammatical and spelling errors and inconsistencies (eg a mixture of verbs and nouns in a list of menu items, lack of consistency in the use of capital letters).

In this respect, it is **not advisable to let developers provide any user interface text items** (since they will generally be misspelled!). They should be delivered by the project owner as early as possible in the development cycle, in order to avoid having developers create temporary elements that might remain "as is" in the final product.

If testing reveals major **problems in terms of ease of use** of the product, due to incomplete or imprecise requirements or poor design, it may be necessary to **negotiate design changes** with the developer, which is generally easier to achieve with an in-house development team…

Functional tests

The purpose of functional testing is to check that **all specified functions and features** of a product (including back-office tools) **have been implemented and work properly**. Tests must be systematically performed for each function and each feature, even the most mundane such as "login", "logout", "copy & paste" and "print".

As mentioned above, in order to check the compliance of the product's functions and features with its specifications, testers need to have access to them or, even better, they should be provided with **precise scenarios** (test scripts) describing the specific tests to be carried out and the expected result of each test.

For certain functions, a **special test version** may be required. For example a test version of an online payment system should be used in order to avoid any actual disbursement on the part of the testers.

Technical tests

Technical tests are intended to check that all of the software's functions and features work properly on all **platforms** (computers, servers, smartphones, tablets, etc.), in all **environments** (operating systems, browsers, plug-ins, etc.) and with all minimum **configurations** documented in the requirements specification.

As mentioned above, exhaustive technical testing, in particular configuration testing, generally requires the involvement of a service provider in addition to internal testers.

Technical tests should be conducted neither too soon nor too late. A good choice might be the **first beta version** of the software. If serious problems are detected, additional rounds of tests should be done on the next beta versions.

Waiting for the first beta version of the software to do exhaustive technical testing does not preclude conducting **partial tests** at an earlier stage, eg with an **alpha version**. It is indeed advisable to **identify and report major technical problems as soon as possible**.

Performance and load tests

For certain applications, it is important to check that performance (speed, response time, throughput, etc.) is acceptable with minimum configurations and maximum loads as documented in the requirements specification: speed of search, speed of display, overall performance of client-server software or of a website with a large number of simultaneous users, etc.

Like technical testing, performance testing may be done on a few configurations as soon as the **first alpha version** of the software is available. Full-scale performance testing should be done with **a beta version** of the product and generally requires the use of a specialized service provider and/or of automated performance and load testing tools.

Documentation tests

A product generally includes documentation (for example a Help feature, an installation and operation manual, a "Read me" file). As any other part of a product, documentation must be tested.

Help text, which may be illustrated with screen shots and/or videos/animations, should be **in phase with the product**. This is one of the reasons why Help testing is necessary, and it also explains why Help content should be **finalized at the latest possible stage** of the development process. The schedule agreed upon by the PM and the developer should therefore reflect the need to integrate the final Help content **just before the "Release candidate"** is produced.

If the product includes a **manual**, it should also be in phase with other components of the product, and should be tested accordingly. Usually, a product's manual has to be **sent to print before the product is finalized**. The manual should therefore be **frozen as late as possible** with respect to the deadline for printing.

As regards the "**Read me**" file, if required, its primary purpose is to inform users of errors in the manual and to provide adequate corrections. It is therefore generally finalized and integrated at the "**last minute**". It should nevertheless be proofread.

Accessibility tests

Some products, in particular websites, are required to be **"accessible" to visually impaired users**. Making a website accessible involves complying with the guidelines of the **Web Accessibility Initiative (WAI)** of the WorldWide Web Consortium (W3C).

In theory, testing the accessibility of a website means checking that W3C/WAI guidelines have been followed by the developer, as well as by the content provider (some of the guidelines apply to content).

In practice, compliance with accessibility guidelines should be checked by visually impaired persons working for a specialized testing service provider or for a university lab or any other organization (which will generally be more than happy to be involved in testing, provided that some form of compensation is offered).
Practical accessibility testing covers at least ease of use, in particular navigation, as well as the website's compatibility with **text-to-Braille** and **text-to-speech** software.

> For **more information on the W3C/WAI guidelines**, see www.w3.org/WAI/.

Free tests

It is advisable to supplement the above-mentioned range of tests with testing performed **by people who are not given any guidelines or specific instructions**. Such "free testers" should simply use the product as if they were "real users".

Free testing does not require any reference to the specifications or any test scripts. Free testers should however be required to **make a note** of anything they may find surprising, abnormal or suspicious in the product's content, user interface, functions & features, behaviour and performance.

Of course, feedback from free testers needs to be analyzed, sorted and consolidated by the PM (or by someone designated by the PM), and any **new bugs** that have been identified as a result of free testing should be **adequately described and entered into the bug reporting system**.

Trainees (interns), future in-house users, if any, as well as Marketing & Sales and Customer Services people may be assigned to or asked to volunteer for free testing.

Problem management

Testing is useless if the **problems** it reveals are not **rigorously documented, reported and solved**.

The PM must therefore set up a **process for problem ("bug") reporting, tracking and resolution**.

Such a process can be implemented with a "**bug management**" tool such as Mantis Bug Tracker (see www.mantisbt.org/) installed on an intranet or extranet site.

A bug management tool typically features a data base of "**issues**" and related functions. Each issue is documented in a record with an identifier, a title, a category, a description, a severity level, a priority level, and, if necessary, attached files (eg screenshots).
The person who reports an issue (the "reporter") is also identified.
Of course, the status of each issue changes as work on its resolution progresses (from "new" to "closed", with intermediate stages).
The PM, the development subproject manager, the developers, the testers and any other persons that have been given access to the bug management system may be included in a distribution list for e-mails that are automatically sent by the system whenever a new issue is created and/or when an existing issue record is changed.

Each **issue** should be described in a **clear and precise** fashion. The description should include instructions that enable the developer to **reproduce the problem**. If it is possible, and useful, the problem should be **illustrated** by one or several screenshots.

I strongly advise to deal with **one bug at a time**, ie an issue record should document a **single problem** only. Indeed, experience shows that if a record addresses several issues, there is a high risk that the person assigned to the resolution of the issues will deal with only one of them, solve it and prematurely mark the issue as being "resolved"!

A **discussion** between a developer and a tester (and/or the PM) relating to a specific issue may proceed by means of **notes** added to the issue record.

When an issue has been marked "resolved" by the developer, the PM should test (or have someone test) that it is **really resolved** before accepting the solution and closing the corresponding data base record.

Records marked "closed" are not physically deleted from the data base, so they remain accessible, which is all the more useful as **bugs that have been fixed** in a given version of the software **may reappear** in a subsequent version. In such a case, the issue record needs to be reopened and resubmitted to the person supposed to have resolved it (and/or to his/her manager…).

Note that **regressions** are unfortunately frequent and generally unavoidable. They make testing even more tedious and costly than it should be (in an ideal world…). It is however a fact that must be accepted, so the PM (and the team of testers) should watch out for regressions.

Involvement of the project manager

The **PM** is **responsible** for the **quality** of the product, so he should be directly and fully involved in the testing and problem management processes. With very large projects however, the PM may need to **delegate** some of his responsibilities to a **Testing subproject manager**. The PM should nevertheless get involved at some stage of the process, if only to check that testing has been properly planned and carried out seriously, which may require some testing by the PM!

The PM or the Testing subproject manager may decide to **let testers document issues directly** in the bug tracking system. In this case, testers should be given precise guidelines, and it is necessary to check that the guidelines are actually followed.
The PM or the Testing subproject manager may however prefer to **centralize the collection of bug descriptions** written by testers, in order to review them, to eliminate duplicates and, if necessary, to rewrite the descriptions before creating new records in the data base. This approach is generally necessary in the case of free testing and may also be applied to the findings resulting from tests performed by service providers.

> *At Hachette, I made a point of getting deeply involved in testing, particularly for major reference products such as the encyclopedia and dictionaries.*

> *In fact, I would not rest until I was confident that the product had reached a high level of quality, which required total dedication to the task, including spending the better part of weekends performing tests!*
> *Furthermore, as far as the EHM was concerned, I was in charge of entering issues into the bug management system and coordinating their resolution in close cooperation with the development team.*
> *By the way, at one point in the development of a new version of the EHM, the number of software bugs that had been identified exceeded 1,024 and, to dramatize the event, I sent a congratulations message to the development team, which some developers did not really appreciate!*

Product acceptance

Acceptance of a product is a process which is usually applied to the successive versions of the product under development (alpha, beta… final). If the result of testing a given version of the product is positive, then that version is accepted, otherwise it is rejected! The above applies to a prototype of a product as well as to the "real product".

Testing performed on behalf of the project owner is sometimes called "**user acceptance testing (UAT)**", as opposed to testing carried out by developers.

As mentioned in Chapter 8 ("Requirements specification"), in the case of business applications and, particularly, for complex information systems (IS), specific testing phases may be required and therefore specified by the project owner, such as:

- an "**operational acceptance testing (OAT)**" phase, sometimes called "**operational readiness testing**", intended to determine whether the product meets requirements in a real-life situation on a limited scale before it is fully deployed;

- an "**operational health check (OHC)**", intended to verify that the product meets requirements after being fully deployed and put into unrestricted and regular service.

As the deadline for completion of the development work gets closer, it may become apparent that some of the bugs identified as a result of testing cannot be fixed without delaying the product. In such a situation, the PM must exercise his judgment, weigh the "pros and cons", and decide either to give the developer a little more time or to release the product with "**known bugs**".

In principle, known bugs should not be promoted to the status of "feature" (applying the principle described by "if you can't fix it, feature it!" is certainly not good practice…). The problems should be fixed for a subsequent version of the product, in agreement with the developer, which may require some form of negotiation and agreement preferably reached before the product is officially accepted in its current state.

Once the tests have been completed and a decision has been made concerning any known bugs, it is **the PM's responsibility** to promote the final "Release Candidate" to the status of "Final product"!
Depending on the context (organization, level of authority of the PM, etc.), a representative of the **project owner** other than or in addition to the PM (and in addition to the Testing subproject manager, if any) may need to be **involved in the final acceptance process**, which should be featured as an explicit task in a project plan.

> *At Hachette, as director of a Products division, I had been given authority and full responsibility for the acceptance of products managed within my division. So if I declared a product worthy of being released, then it was released. In some cases however, I felt it was necessary to obtain prior approval from my management, which usually involved explaining the situation and justifying the fact that there was more to gain from releasing the product than from delaying it.*

17) Project direction/supervision – Monitoring and control

General remarks

The major role of a PM beyond the project planning phase consists in **directing project execution**, ie **supervising the implementation of the project plan**.

Project direction involves ensuring that the **project** is **executed** with the highest possible degree of **compliance with** its **"baseline" scope, schedule and budget**, as well as with the **quality requirements** for its deliverables.
Project supervision involves **monitoring and controlling all of the tasks** performed by the project team and contractors.

According to the scale of the project and the type of project organization, the PM may be able to **delegate some of the supervision work to subproject managers** or he may have to supervise all of the project tasks himself.
However that may be, the PM's role in **directing the project is comparable to conducting an orchestra**, ensuring that all players execute the score in harmony.

Monitoring and controlling the project require **processes** for **gathering information,** as well as **tracking and reviewing progress** in each of the project's areas. These processes should enable the PM to **identify deviations from the plan** as well as **problems, issues and risks**, which should lead the PM, whenever necessary, to take **corrective action** without delay.

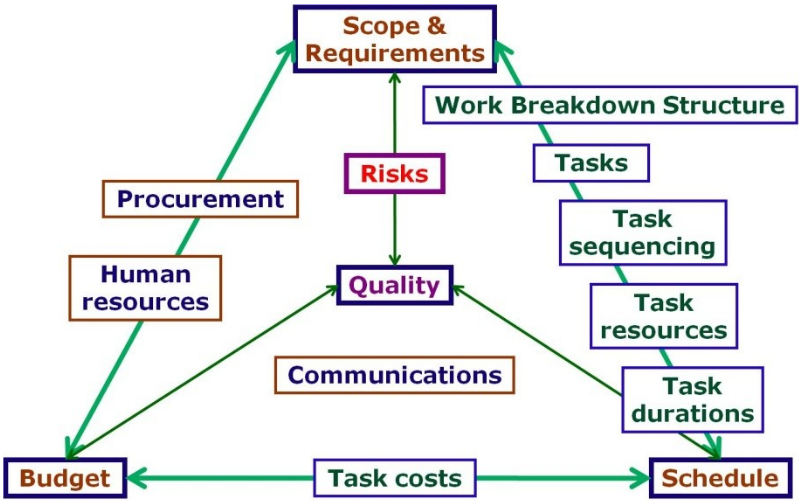
Note that "Murphy's law", as applied to project management, generally proves to be true:

- *"Anything that can go wrong will go wrong."*

> For **more information on this subject**, see en.wikipedia.org/wiki/Murphy%27s_law.

The PM must therefore permanently **keep a lookout for problems** in all areas of the project.

The **areas to be monitored and controlled** are the same as those addressed in the project management plan, as summarized in the following "expanded triangle" diagram.



Note that a project plan is rarely perfect, so **deviation from the plan is normal**. Deviation may actually be necessary. It is sometimes even beneficial.

Major changes with respect to the plan, in particular concerning scope, schedule and budget, not only need to be documented (in a "**change log**"), but also usually need to be formally **authorized by the project owner and/or sponsor**.

Requirements may need to be changed as a result of the design process or due to technical difficulties encountered by the developers, or for any other valid reason.

Flexibility as regards changes to the requirements is desirable, but requirements must be "frozen" at some point in order to **avoid** the disadvantages of a "**moving target**" or of "**scope creep**" (ie **uncontrolled expansion** of a project's scope), unless circumstances allow the PM to apply "agile" principles all the way (as explained in chapter 12).

The **work breakdown structure** (WBS) should be used by the PM as a reference document that provides an overview of the project's activities/tasks (grouped into work packages). It may also be used as a framework for progress reporting purposes.
The WBS may need to be amended as work progresses to take into account changes that may be necessary in the overall organization of the project.

Tasks (or activities) may be proceeding less smoothly than expected. Difficulties need to be analyzed and resolved. Solutions need to be devised in order to remove or by-pass obstacles. The PM may also establish that certain tasks that had not been planned need to be added to a work package.

There may be problems with the **sequencing of tasks**. For example, new dependencies between tasks may appear, which may have a direct negative impact on "successor" tasks if adequate action is not taken early enough. And any new tasks that are brought into the picture need to be fitted into the project's sequenced list of tasks at appropriate places in relation to other tasks.

Resources assigned to a given task may turn out to be insufficient in order to complete it on schedule while maintaining the required level of quality, in which case additional resources must be allocated to the task or its duration must be extended.
On the other hand, it may become apparent at some stage that completing a task requires fewer resources than estimated. In that case, some of the resources may be released or reassigned earlier than planned.
As mentioned above, it may be necessary to add tasks to a work package, in which case adequate resources need to be assigned to those new tasks.

The **duration** of certain tasks may be longer or shorter than planned, which may have a negative or positive impact on the overall **schedule**.

The **cost** of certain tasks may be higher than estimated, due to their extended duration or the actual cost of resources or the addition of resources.
On the other hand, there may be good surprises! A lower-than-estimated actual cost of certain tasks may counterbalance a cost overrun of others, thus allowing the project to remain within **budget**.

Here is an example of a project for which a major change in scope was decided in order to make optimum use of the remainder of the budget, at a time when one third of it had already been spent.

> *Hachette Multimedia participated, from June 2002 to November 2004, in a large European project ("CELEBRATE") subsidized by the European Commission, which consisted in designing and producing a hundred or so "Learning Objects (LOs)" for schools, in English and in French. The LOs were fairly sophisticated animations, developed in Flash, illustrating principles and facts in physics and natural sciences. Each LO required a script, specific graphic design work and software development. The cost of those custom-made LOs was relatively high and it soon became obvious that the objective in terms of number of LOs to be delivered could not be achieved. So, in agreement with the overall project coordinator at the European level, I (the PM at Hachette) decided to spend the remaining two thirds of the development budget on the creation of a series of "LO templates" that would allow editors to produce a large number of LOs on a wide variety of subjects, without any further need for graphic designers or developers.*

In the area of **Human Resources**, the PM may encounter problems implementing the **hiring plan**, for example a person who was supposed to join the team at a certain date may have decided to accept another job. In that case, **emergency action** needs to be taken in order to find a suitable replacement.
There may also be **conflicts** between members of the team that need **to be resolved**.

Solutions to problems encountered by any given team member should be **communicated** to other team members who are liable to face the same problem. In that respect, a **knowledge base or "FAQ"** available on the project's intranet site is usually very helpful.

As regards **procurement**, contracts and purchase orders must be signed and executed according to the plan. Any problem that may arise in that area must be solved. For example, a given supplier may not be able to provide goods or services as expected, or may demand a higher compensation than estimated or initially agreed upon.

Risks identified during the planning phase of the project may materialize and new risks may appear. They should be dealt with appropriately, ie their **impact** needs to be **evaluated** and **action** should be **taken** in order to **mitigate or eliminate** them.

The PM should closely monitor and control the **quality of project execution**, as well as the **quality of its deliverables**, in particular the **resulting product** in its successive stages of development.

In the area of **communications**, the PM must ensure that **adequate information** is provided in a timely fashion **to those who "need to know"**, including subproject managers, if any, team members, contractors, management, the project owner and sponsor, Marketing & Sales, etc.
In particular, the PM must make sure that **team members** have **easy access to** all of the **information** they need to do their jobs.

The PM, who is ultimately responsible for the outcome of the project, must also make sure that all of the **information needed to monitor and control the project** is available to him when required. No essential information should be overlooked (nor hidden!), in particular concerning **serious issues** that have not been resolved or **major risks** that may have materialized.

For that purpose, the PM must make sure that subproject managers, if any, or team members themselves provide information on a regular basis and **escalate important issues** as soon as they appear.

Project supervision requires above all an **excellent organization**, with **well-defined hierarchical or functional relationships**, and a **well-oiled mechanism for information flow**, hence the **paramount importance of good communications**.

Meetings

The PM should review the progress of the project on the occasion of **regular one-to-one or group meetings** with subproject managers, if any, or directly with selected team members.

Group meetings provide each participant with the opportunity to report on the progress of the tasks he/she is in charge of, to point out problems, to raise issues, and to gain awareness of the situation of other tasks and of the project as a whole. Problems and proposed solutions may be discussed by the group. A group meeting may feature a "**brainstorming session**" to work out **solutions to complex problems**.
After every meeting of importance, the PM should publish a **report**, highlighting **decisions** and **action items**.

For each action item, the PM should specify a **level of priority**, the **deadline** by which it has to be completed and the name of the **person** it has been **assigned** to.

Action items determined at a meeting should be **reviewed** during the next meeting (or earlier if necessary), in order to check that execution has at least started, that it is moving in the right direction and that any visible results meet expectations.

Although most meetings are absolutely necessary, the PM should make sure that they do not consume an excessive amount of time. The PM should **carefully prepare each meeting**, set an **agenda** and a **schedule**, and take care to keep **discussions within scope** (unless there is a good reason to expand the scope…).

- Meetings should <u>not</u> be "events where minutes are taken and hours are lost"!

It is also desirable to hold meetings from time to time with the whole of the **project team**, to give all team members an overview of the project's status, and to let each person express his/her thoughts and possibly discuss them with the group.

Project team meetings may be held for example once a month. The PM should make a short presentation of the **overall project's progress**. Subproject managers, if any, should then each make a presentation concerning their specific part of the project.
If there is more to show than just slides, for example a version of the product being developed, then a **demonstration** should be featured on the meeting agenda.
The meeting schedule should provide enough time for a **questions & answers** session.

Project team meetings are useful not only for delivering information to team members, but also for gathering their **feedback and suggestions**. Those meetings also provide the PM with additional opportunities to **mobilize and motivate** the team.

For some projects, or during certain phases of a project, it may be necessary to hold meetings with subproject managers, if any, or with the whole project team on a very frequent basis, for example daily. Such meetings should be kept **short and focused**.
An effective form of meeting that satisfies those conditions is the so-called **"standup meeting"**: participants get together in the manager's office or in a meeting room or in the team's work area, and whatever is presented and/or discussed is done with everyone standing, not sitting.

Meetings may obviously also be called by the PM on an **ad hoc** basis, as the need arises.

Regular meetings with **major contractors** (such as a software developer) are also necessary. The PM may delegate the appropriate subproject manager(s) to such meetings, but some of them may require the PM's presence, for example meetings where requirements or functional design specifications are discussed, or meetings where a "milestone" deliverable, eg a version of the product under development, is reviewed.
The above-mentioned **general guidelines for meetings** also apply to meetings with contractors, in particular: **careful preparation, predetermined agenda and schedule, keeping discussions within scope**. Each meeting should be followed by a **meeting report with detailed action items**.

Finally, the PM is usually required to report to his **management** (and/or project owner/sponsor) on the **progress of the project** on a regular basis, for example once a month. **Reporting** may be done on the occasion of more or less formal meetings where the PM and subproject managers, if any, each deliver a presentation. Whenever possible, management should be shown the latest version of the product under development.

The PM must prepare management-level project review meetings with great care in order to be in a position to provide **satisfactory answers** to questions from his **management** and his **financial controller**. Participants generally expect to receive handouts, including a summary of where the project stands with respect to work package completion, schedule and budget, which can be provided as a set of performance indicators (as described further on in this chapter).

Management By Walking Around (MBWA)

MBWA is a **management technique** that is very effective **for gathering information**, for "gauging the temperature", as well as for motivating staff, in a relatively informal fashion. It consists for a manager in spending time, as frequently as he feels right or necessary, with people in his organization, at their place of work as opposed to meeting with them in his own office.

A PM should **practice MBWA on a regular basis**. He should pay visits not only to subproject managers but also to all of the team members (though maybe not all at once…). The PM should however be careful to **avoid disrupting work** in progress.

A **direct contact with team members** enables the PM to **ask questions**, to **keep abreast of what is going on** in the project team, to **feel the atmosphere**, to **detect difficulties**, situations of **conflict**, **dissatisfaction**, etc.
MBWA also provides the PM an opportunity to **give information** to team members, to **answer** their **questions**, to **motivate** them (**encourage** them and maybe even **congratulate** them for a "job well done"!).

The PM should however not bypass subproject managers when it comes to managing their respective teams on a day-to-day basis. He should also avoid delivering information or instructions that subproject managers are not aware of, or that might contradict information or instructions already given.

MBWA is generally appreciated by team members if it conforms to **rules of good practice** and visibly contributes to the smooth running of the project, as well as to a good atmosphere in the project team.

Here is a link to a **short article on the subject**: www.futurecents.com/mainmbwa.htm.

Performance indicators

Information gathered in the process of monitoring the progress of activities in the various project areas must be **analyzed, sorted and consolidated**. Important information should be transcribed into written form, in particular meeting reports. It may be used to determine immediate action items or stored for future reference. It may be **summarized** for progress reports and used to update **performance indicators**, which may sometimes be limited to a set of **key performance indicators ("KPIs")**.

Performance indicators should be designed in order to provide **a summary view of the status** ("health") **of a project** and a **measurement of its progress in each of its key areas**, in **quantitative** as well as **qualitative** terms.

The format of performance indicators used by **subproject** managers for their respective work packages should be as **compatible** as possible with the format of indicators for the **overall project**, so as to facilitate **consolidation** by the PM.

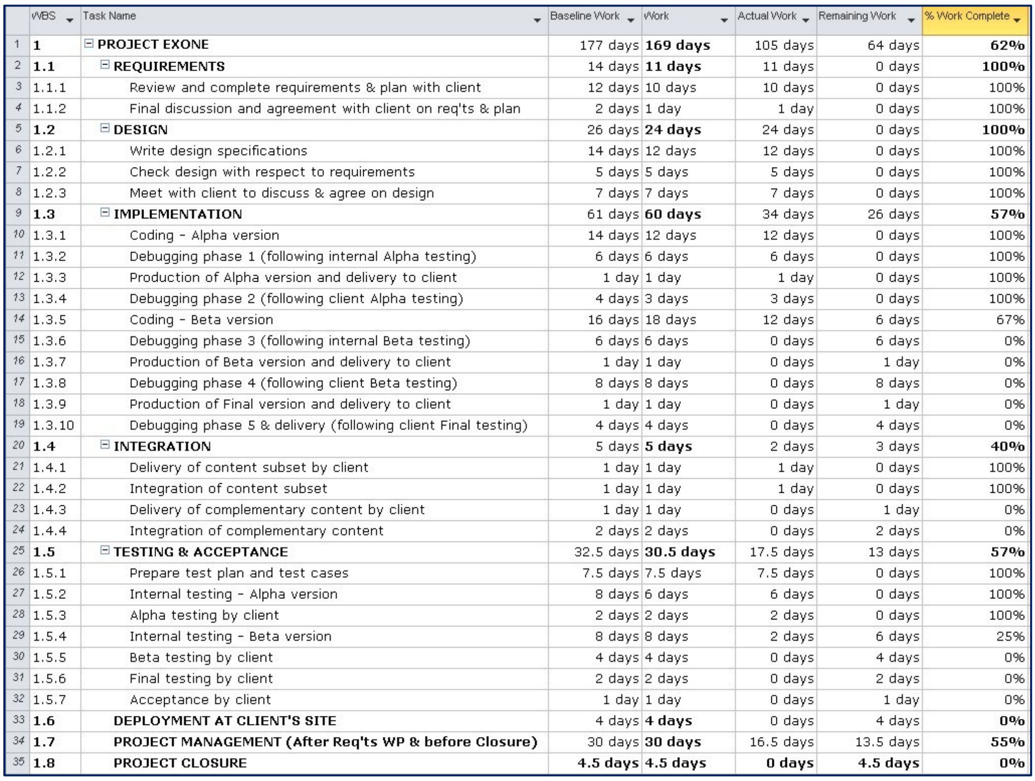
Here is a **non-exhaustive list of KPIs** that should be kept up-to-date throughout the project's execution phase:

- degree of completion of work packages and tasks,
- status of deliverables with respect to target milestones,
- status of the project with respect to its baseline schedule,
- evolution of cost (spending) with respect to the budget,
- assessment of remaining risks.

Quantitative **indicators in tabular form should also be represented graphically**, whenever relevant and possible, in order to increase their legibility and to make differences (eg "achieved vs planned") appear more evidently.

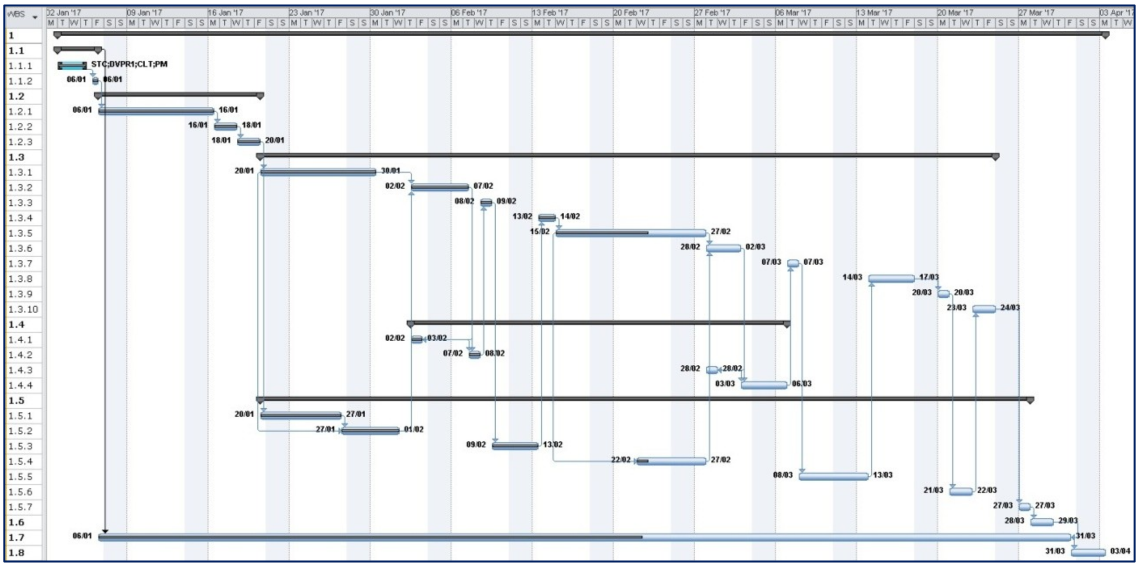
A few **examples of performance indicators** are given below. Some were created with Microsoft Project, others with Microsoft Excel (which in many cases is easier to use than more specialized software to create and update performance indicators).

Here is a tabular view (produced with MS Project) of the progress of the contractor's work on project EXONE at a given date, showing for each task the amount of work, expressed in person-days, initially estimated ("Baseline Work" column), re-evaluated ("Work" column), actually done ("Actual Work" column) and remaining to be done ("Remaining Work" column); the rightmost column ("% Work Complete") shows the percentage of completion of each task and work package, and of the project as a whole.



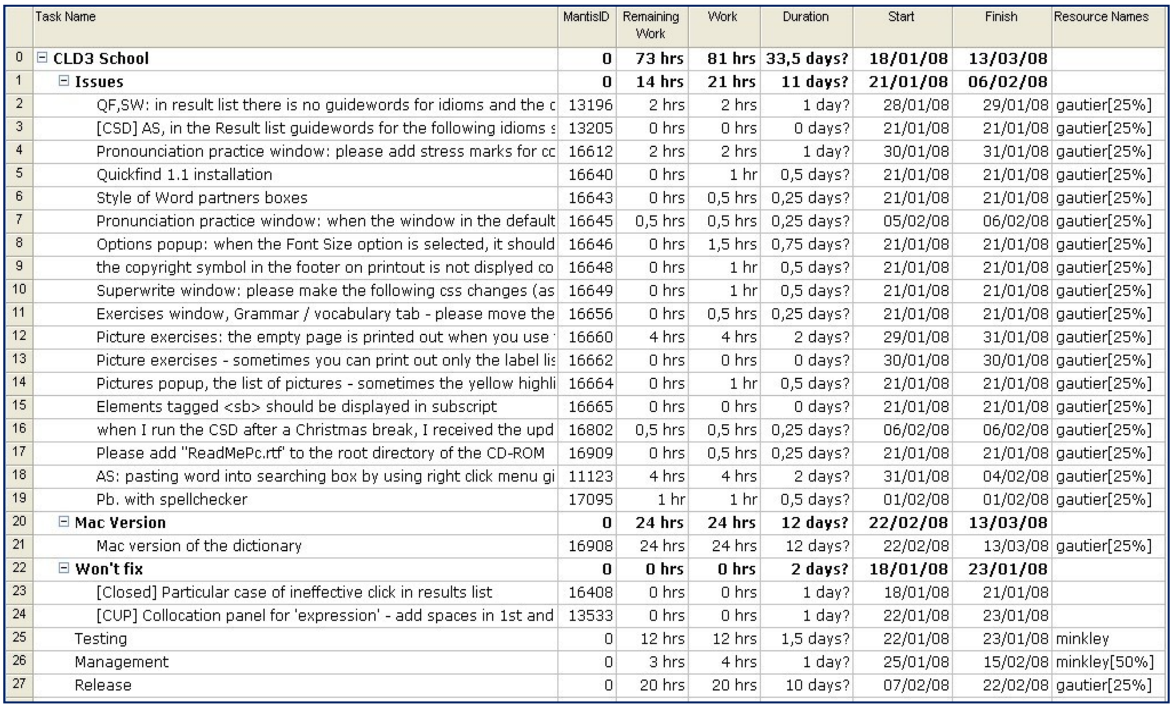
	WBS	Task Name	Baseline Work	Work	Actual Work	Remaining Work	% Work Complete
1	**1**	⊟ **PROJECT EXONE**	177 days	**169 days**	105 days	64 days	**62%**
2	**1.1**	⊟ **REQUIREMENTS**	14 days	**11 days**	11 days	0 days	**100%**
3	1.1.1	Review and complete requirements & plan with client	12 days	10 days	10 days	0 days	100%
4	1.1.2	Final discussion and agreement with client on req'ts & plan	2 days	1 day	1 day	0 days	100%
5	**1.2**	⊟ **DESIGN**	26 days	**24 days**	24 days	0 days	**100%**
6	1.2.1	Write design specifications	14 days	12 days	12 days	0 days	100%
7	1.2.2	Check design with respect to requirements	5 days	5 days	5 days	0 days	100%
8	1.2.3	Meet with client to discuss & agree on design	7 days	7 days	7 days	0 days	100%
9	**1.3**	⊟ **IMPLEMENTATION**	61 days	**60 days**	34 days	26 days	**57%**
10	1.3.1	Coding - Alpha version	14 days	12 days	12 days	0 days	100%
11	1.3.2	Debugging phase 1 (following internal Alpha testing)	6 days	6 days	6 days	0 days	100%
12	1.3.3	Production of Alpha version and delivery to client	1 day	1 day	1 day	0 days	100%
13	1.3.4	Debugging phase 2 (following client Alpha testing)	4 days	3 days	3 days	0 days	100%
14	1.3.5	Coding - Beta version	16 days	18 days	12 days	6 days	67%
15	1.3.6	Debugging phase 3 (following internal Beta testing)	6 days	6 days	0 days	6 days	0%
16	1.3.7	Production of Beta version and delivery to client	1 day	1 day	0 days	1 day	0%
17	1.3.8	Debugging phase 4 (following client Beta testing)	8 days	8 days	0 days	8 days	0%
18	1.3.9	Production of Final version and delivery to client	1 day	1 day	0 days	1 day	0%
19	1.3.10	Debugging phase 5 & delivery (following client Final testing)	4 days	4 days	0 days	4 days	0%
20	**1.4**	⊟ **INTEGRATION**	5 days	**5 days**	2 days	3 days	**40%**
21	1.4.1	Delivery of content subset by client	1 day	1 day	1 day	0 days	100%
22	1.4.2	Integration of content subset	1 day	1 day	1 day	0 days	100%
23	1.4.3	Delivery of complementary content by client	1 day	1 day	0 days	1 day	0%
24	1.4.4	Integration of complementary content	2 days	2 days	0 days	2 days	0%
25	**1.5**	⊟ **TESTING & ACCEPTANCE**	32.5 days	**30.5 days**	17.5 days	13 days	**57%**
26	1.5.1	Prepare test plan and test cases	7.5 days	7.5 days	7.5 days	0 days	100%
27	1.5.2	Internal testing - Alpha version	8 days	6 days	6 days	0 days	100%
28	1.5.3	Alpha testing by client	2 days	2 days	2 days	0 days	100%
29	1.5.4	Internal testing - Beta version	8 days	8 days	2 days	6 days	25%
30	1.5.5	Beta testing by client	4 days	4 days	0 days	4 days	0%
31	1.5.6	Final testing by client	2 days	2 days	0 days	2 days	0%
32	1.5.7	Acceptance by client	1 day	1 day	0 days	1 day	0%
33	**1.6**	**DEPLOYMENT AT CLIENT'S SITE**	4 days	**4 days**	0 days	4 days	**0%**
34	**1.7**	**PROJECT MANAGEMENT (After Req'ts WP & before Closure)**	30 days	**30 days**	16.5 days	13.5 days	**55%**
35	**1.8**	**PROJECT CLOSURE**	**4.5 days**	**4.5 days**	**0 days**	**4.5 days**	**0%**

Here is the corresponding graphical view (the lighter-coloured bars or parts of bars represent work remaining to be done).



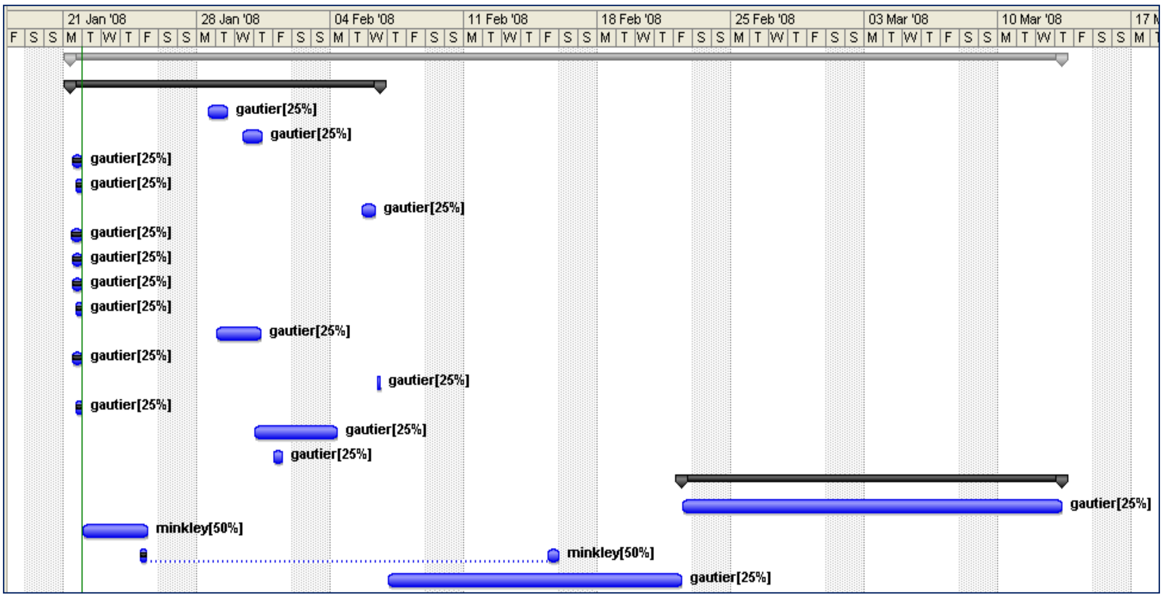
Various more or less complete views of work progress may be used.
The following (real-life) example concerns the final stage of a development effort (for an online English dictionary).

Microsoft Project was used to track the amount of work remaining for bug fixing, testing and software release production, as well as to determine a schedule given the availability of the two persons involved (in particular, the software development engineer was available only 25% of his time for this project).



	Task Name	MantisID	Remaining Work	Work	Duration	Start	Finish	Resource Names
0	⊟ **CLD3 School**	**0**	**73 hrs**	**81 hrs**	**33,5 days?**	**18/01/08**	**13/03/08**	
1	⊟ **Issues**	**0**	**14 hrs**	**21 hrs**	**11 days?**	**21/01/08**	**06/02/08**	
2	QF,SW: in result list there is no guidewords for idioms and the c	13196	2 hrs	2 hrs	1 day?	28/01/08	29/01/08	gautier[25%]
3	[CSD] AS, in the Result list guidewords for the following idioms s	13205	0 hrs	0 hrs	0 days?	21/01/08	21/01/08	gautier[25%]
4	Pronounciation practice window: please add stress marks for cc	16612	2 hrs	2 hrs	1 day?	30/01/08	31/01/08	gautier[25%]
5	Quickfind 1.1 installation	16640	0 hrs	1 hr	0,5 days?	21/01/08	21/01/08	gautier[25%]
6	Style of Word partners boxes	16643	0 hrs	0,5 hrs	0,25 days?	21/01/08	21/01/08	gautier[25%]
7	Pronunciation practice window: when the window in the default	16645	0,5 hrs	0,5 hrs	0,25 days?	05/02/08	06/02/08	gautier[25%]
8	Options popup: when the Font Size option is selected, it should	16646	0 hrs	1,5 hrs	0,75 days?	21/01/08	21/01/08	gautier[25%]
9	the copyright symbol in the footer on printout is not displyed co	16648	0 hrs	1 hr	0,5 days?	21/01/08	21/01/08	gautier[25%]
10	Superwrite window: please make the following css changes (as	16649	0 hrs	1 hr	0,5 days?	21/01/08	21/01/08	gautier[25%]
11	Exercises window, Grammar / vocabulary tab - please move the	16656	0 hrs	0,5 hrs	0,25 days?	21/01/08	21/01/08	gautier[25%]
12	Picture exercises: the empty page is printed out when you use	16660	4 hrs	4 hrs	2 days?	29/01/08	31/01/08	gautier[25%]
13	Picture exercises - sometimes you can print out only the label lis	16662	0 hrs	0 hrs	0 days?	30/01/08	30/01/08	gautier[25%]
14	Pictures popup, the list of pictures - sometimes the yellow highli	16664	0 hrs	1 hr	0,5 days?	21/01/08	21/01/08	gautier[25%]
15	Elements tagged <sb> should be displayed in subscript	16665	0 hrs	0 hrs	0 days?	21/01/08	21/01/08	gautier[25%]
16	when I run the CSD after a Christmas break, I received the upd	16802	0,5 hrs	0,5 hrs	0,25 days?	06/02/08	06/02/08	gautier[25%]
17	Please add "ReadMePc.rtf' to the root directory of the CD-ROM	16909	0 hrs	0,5 hrs	0,25 days?	21/01/08	21/01/08	gautier[25%]
18	AS: pasting word into searching box by using right click menu gi	11123	4 hrs	4 hrs	2 days?	31/01/08	04/02/08	gautier[25%]
19	Pb. with spellchecker	17095	1 hr	1 hr	0,5 days?	01/02/08	01/02/08	gautier[25%]
20	⊟ **Mac Version**	**0**	**24 hrs**	**24 hrs**	**12 days?**	**22/02/08**	**13/03/08**	
21	Mac version of the dictionary	16908	24 hrs	24 hrs	12 days?	22/02/08	13/03/08	gautier[25%]
22	⊟ **Won't fix**	**0**	**0 hrs**	**0 hrs**	**2 days?**	**18/01/08**	**23/01/08**	
23	[Closed] Particular case of ineffective click in results list	16408	0 hrs	0 hrs	1 day?	18/01/08	21/01/08	
24	[CUP] Collocation panel for 'expression' - add spaces in 1st and	13533	0 hrs	0 hrs	1 day?	22/01/08	23/01/08	
25	Testing	0	12 hrs	12 hrs	1,5 days?	22/01/08	23/01/08	minkley
26	Management	0	3 hrs	4 hrs	1 day?	25/01/08	15/02/08	minkley[50%]
27	Release	0	20 hrs	20 hrs	10 days?	07/02/08	22/02/08	gautier[25%]

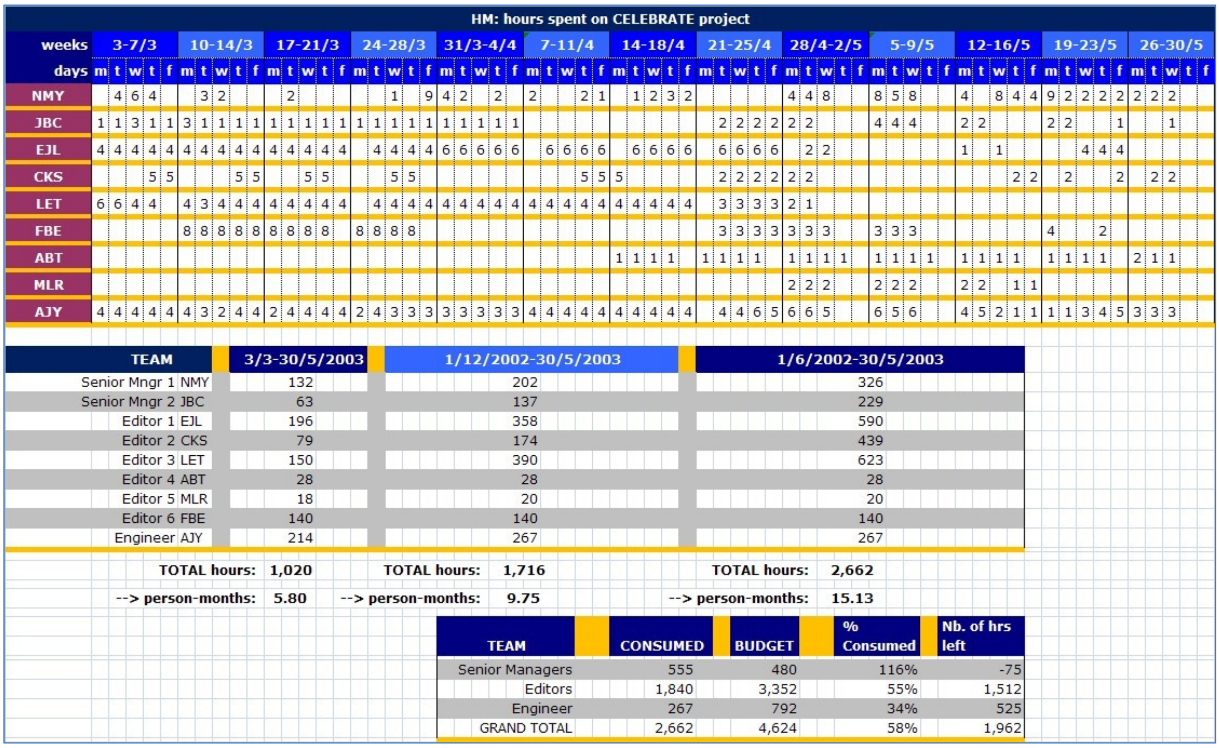
The following Gantt chart corresponds to the previous tabular view.



The next (real-life) example shows part of a “time sheet” that was used to record work (in hours per day) performed by the various persons involved in the previously-mentioned “CELEBRATE” project over a certain period, as well as the total number of hours spent by each person, firstly during the current period, secondly during the most recent six-month period, and thirdly since the beginning of the project.

It also provides a comparison between total “consumption” and budget (in hours), broken down into categories of personnel.

This level of detail was necessary for the calculation of costs to be reported to the European Commission, which subsidized the project.



HM: hours spent on CELEBRATE project

weeks	3-7/3					10-14/3					17-21/3					24-28/3					31/3-4/4				
days	m	t	w	t	f	m	t	w	t	f	m	t	w	t	f	m	t	w	t	f	m	t	w	t	f
NMY		4	6	4			3	2				2						1		9	4	2		2	
JBC	1	1	3	1	1	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
EJL	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4		4	4	4	4	6	6	6	6	6
CKS				5	5				5	5			5	5				5	5						
LET	6	6	4	4		4	3	4	4	4	4	4	4	4	4		4	4	4	4	4	4	4	4	4
FBE						8	8	8	8	8	8	8	8	8		8	8	8	8						
ABT																									
MLR																									
AJY	4	4	4	4	4	4	3	2	4	4	2	4	4	4	4	2	4	3	3	3	3	3	3	3	3

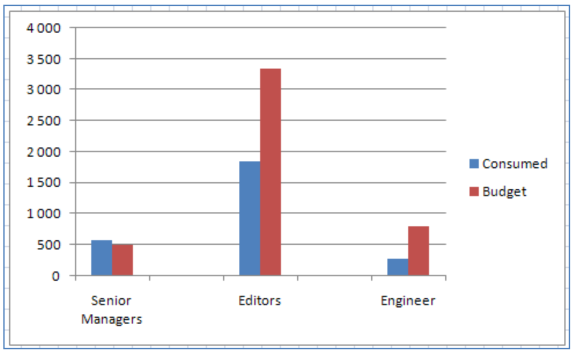
weeks	7-11/4					14-18/4					21-25/4					28/4-2/5				
days	m	t	w	t	f	m	t	w	t	f	m	t	w	t	f	m	t	w	t	f
NMY	2			2	1		1	2	3	2						4	4	8		
JBC												2	2	2	2	2	2			
EJL		6	6	6	6		6	6	6	6		6	6	6	6		2	2		
CKS				5	5	5						2	2	2	2	2	2			
LET	4	4	4	4	4	4	4	4	4	4		3	3	3	3	2	1			
FBE												3	3	3	3	3	3	3		
ABT						1	1	1	1		1	1	1	1		1	1	1	1	
MLR																2	2	2		
AJY	4	4	4	4	4	4	4	4	4	4		4	4	6	5	6	6	5		

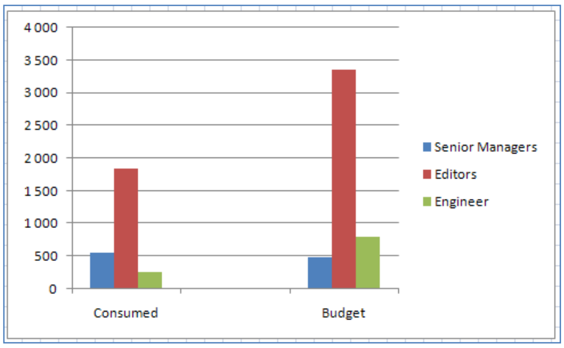
weeks	5-9/5					12-16/5					19-23/5					26-30/5				
days	m	t	w	t	f	m	t	w	t	f	m	t	w	t	f	m	t	w	t	f
NMY	8	5	8			4		8	4	4	9	2	2	2	2	2	2	2		
JBC	4	4	4			2	2				2	2			1			1		
EJL						1		1					4	4	4					
CKS									2	2		2			2		2	2		
LET																				
FBE	3	3	3								4			2						
ABT	1	1	1	1		1	1	1	1		1	1	1	1		2	1	1		
MLR	2	2	2			2	2		1	1										
AJY	6	5	6			4	5	2	1	1	1	1	3	4	5	3	3	3		

TEAM		3/3-30/5/2003	1/12/2002-30/5/2003	1/6/2002-30/5/2003
Senior Mngr 1	NMY	132	202	326
Senior Mngr 2	JBC	63	137	229
Editor 1	EJL	196	358	590
Editor 2	CKS	79	174	439
Editor 3	LET	150	390	623
Editor 4	ABT	28	28	28
Editor 5	MLR	18	20	20
Editor 6	FBE	140	140	140
Engineer	AJY	214	267	267
	TOTAL hours:	1,020	1,716	2,662
	--> person-months:	5.80	9.75	15.13

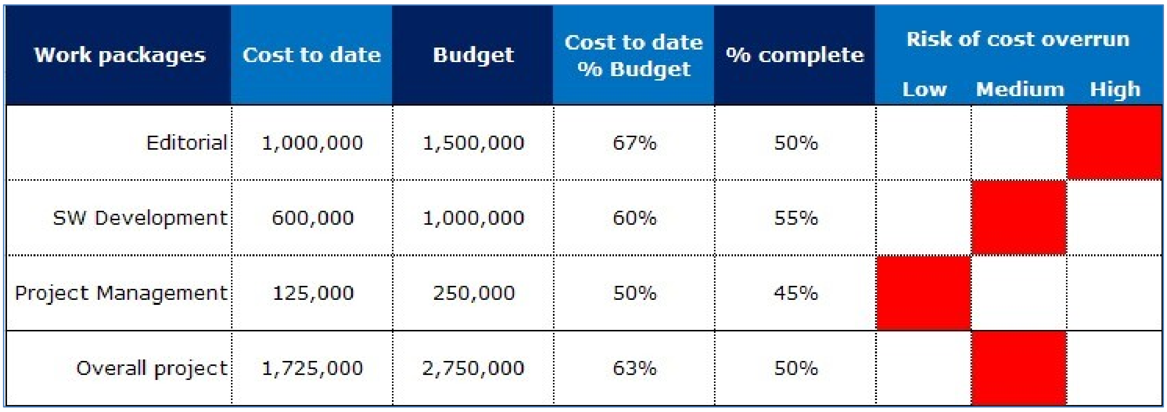
TEAM	CONSUMED	BUDGET	% Consumed	Nb. of hrs left
Senior Managers	555	480	116%	-75
Editors	1,840	3,352	55%	1,512
Engineer	267	792	34%	525
GRAND TOTAL	2,662	4,624	58%	1,962

Here are two graphical representations of the numbers in the third table appearing in the previous illustration.

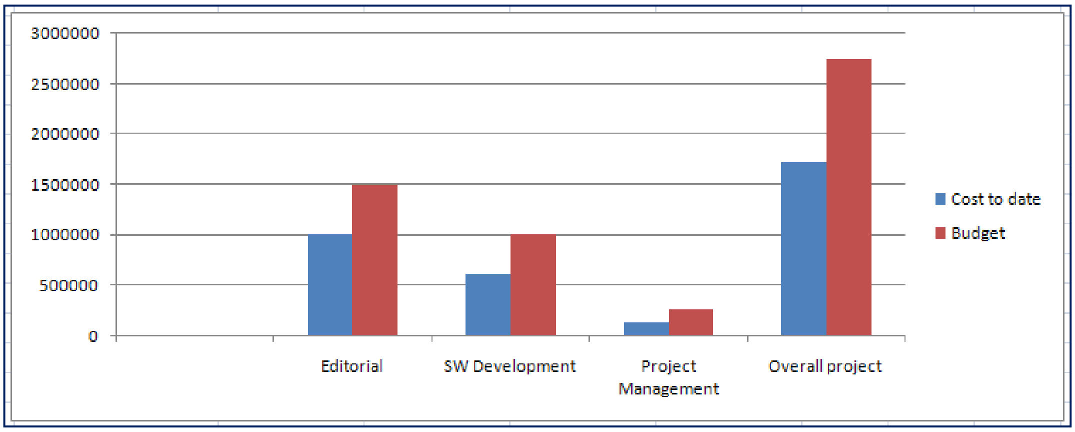




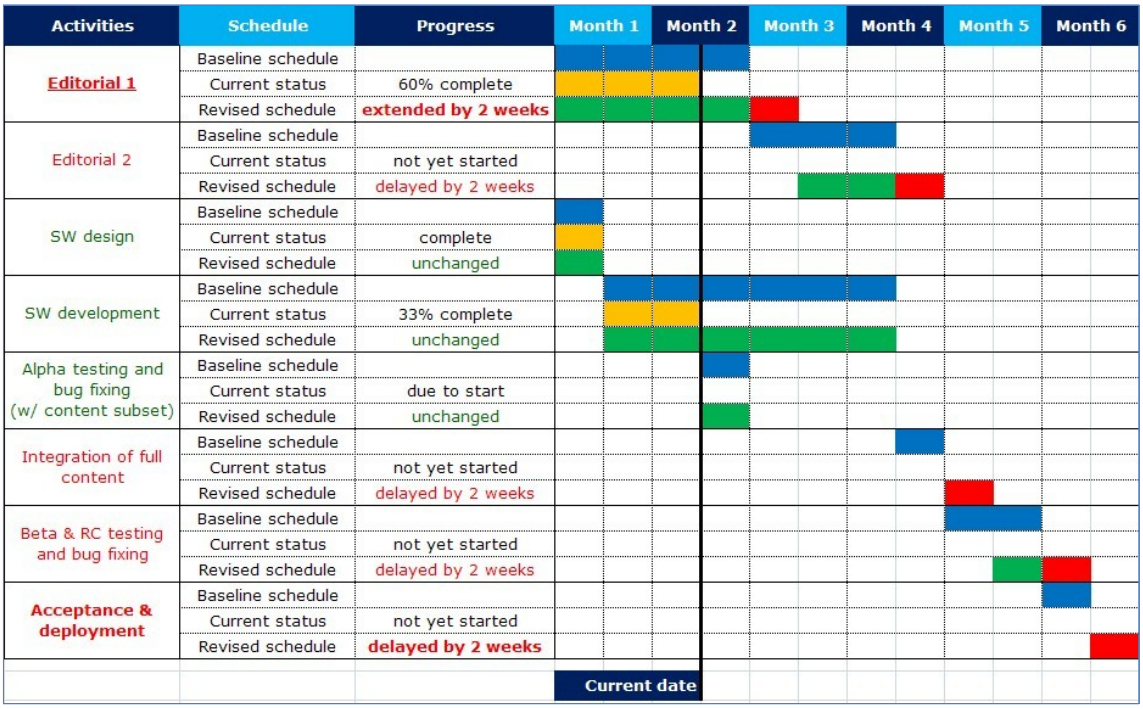
The following example provides tabular and graphical comparisons between "cost to date" and budget for a hypothetical project with three work packages. It also indicates the degree of completion of each WP, and a rough evaluation of the risk of cost overrun.



Work packages	Cost to date	Budget	Cost to date % Budget	% complete	Risk of cost overrun		
					Low	Medium	High
Editorial	1,000,000	1,500,000	67%	50%			🟥
SW Development	600,000	1,000,000	60%	55%		🟥	
Project Management	125,000	250,000	50%	45%	🟥		
Overall project	1,725,000	2,750,000	63%	50%		🟥	



The final example is a performance indicator relating to the progress at a given date ("Current date" = the middle of Month 2) of another hypothetical project with respect to its baseline schedule. The first task should have been 75% complete by that particular date but it is only 60% complete, so its duration has been extended by two weeks in the revised schedule, which has a negative impact of the same order on "successor" tasks.



Activities	Schedule	Progress	Month 1		Month 2		Month 3		Month 4		Month 5		Month 6	
Editorial 1	Baseline schedule		🟦	🟦	🟦	🟦								
	Current status	60% complete	🟨	🟨	🟨									
	Revised schedule	**extended by 2 weeks**	🟩	🟩	🟩	🟩	🟥							
Editorial 2	Baseline schedule						🟦	🟦	🟦					
	Current status	not yet started												
	Revised schedule	delayed by 2 weeks						🟩	🟩	🟥				
SW design	Baseline schedule		🟦											
	Current status	complete	🟨											
	Revised schedule	unchanged	🟩											
SW development	Baseline schedule			🟦	🟦	🟦	🟦	🟦	🟦					
	Current status	33% complete		🟨	🟨									
	Revised schedule	unchanged		🟩	🟩	🟩	🟩	🟩	🟩					
Alpha testing and bug fixing (w/ content subset)	Baseline schedule					🟦								
	Current status	due to start												
	Revised schedule	unchanged				🟩								
Integration of full content	Baseline schedule									🟦				
	Current status	not yet started												
	Revised schedule	delayed by 2 weeks									🟥			
Beta & RC testing and bug fixing	Baseline schedule										🟦	🟦		
	Current status	not yet started												
	Revised schedule	delayed by 2 weeks										🟩	🟥	
Acceptance & deployment	Baseline schedule												🟦	
	Current status	not yet started												
	Revised schedule	**delayed by 2 weeks**												🟥
			Current date											

Project dashboard

The **collection of performance indicators** makes up a **project dashboard**, which is an essential project management tool that the PM should use on a virtually permanent basis for **project direction/supervision** as well as for **reporting** purposes.

Software tools such as a simple spreadsheet or a sophisticated project management application may be used to create and update a project dashboard. The task may be made easier by using ready-made templates, although such templates may not be perfectly adapted to any specific project and may therefore require some adjustment.

There are project dashboard templates (in Excel or HTML format) and even full-blown dashboard applications available on the web (some free of charge, others for a fee): google (or bing) "project dashboard" or "project management dashboard" to find them.

Here is a link to an **interesting article on the subject of project dashboards**:

>> www.anticlue.net/archives/000875.htm

The PM's "To-do list"

The job of a PM involves performing a very large number and a very broad variety of tasks, and it is a **major challenge** for the PM to **make sure that no task is omitted and that all tasks are carried out in a timely fashion**.

A helpful tool for that purpose is a "To-do list". The principle is obvious:

1) make a note of things **to do**,
2) **do** them,
3) mark them as "**done**"!

Of course, in order to be effective, this simple system requires a list of **action items** ("things to do") that is **exhaustive** and kept **up-to-date**.

Action items may result from information gathered in meetings, conversations (over the phone or face to face), e-mails, etc. Another major source of action items is the project dashboard, which the PM should always keep an eye on. In addition, the PM can generate action items just by thinking about the project, which he should be doing all the time!

As mentioned in the "Meetings" section of this chapter, each action item should be given a **level of priority** depending on its importance and degree of urgency, a **deadline for completion**, and, if the PM cannot handle it himself, he should **assign someone** to the action item and hold that person **responsible** for it.

Action items that are both **important and urgent** should be highlighted (and performed ASAP…).
Less important and less urgent tasks should however not be neglected or systematically postponed. The PM must make sure to allocate a certain amount of time every day for such tasks, in order to avoid a build-up that might become unmanageable.

The To-do list should be **updated** as frequently as necessary. The PM should browse through it first thing every morning and plan his day accordingly.

The PM should have his **To-do list handy at any time** and wherever he may be. It can be kept in a notebook, a laptop computer, a smartphone, a phablet or tablet… whatever works!
Adhesive notes should be avoided for obvious reasons (if you think about it…), and using a web-based ("cloud") application is not advisable since access to the Internet may not be available when you need access to your To-do list.

18) Relationship with Marketing & Sales (“M&S”)

General remarks

The Marketing & Sales (“M&S”) department (sometimes two separate departments) of a company takes care of **promoting and selling products**. M&S is therefore a **major stakeholder** in any project whose outcome is a “commercial” product or even a free-of-charge website (since such a site contributes to the company’s image, of which Marketing is usually the guarantor).

As the **primary interface** between the project team and M&S, **the PM** should ensure that M&S is involved at appropriate stages of projects of the kind mentioned above.

The **success of a project** is tied to the success of the resulting product, which generally requires **close and efficient cooperation between the PM and M&S**.

The PM may deal with several people in M&S, in particular with his counterpart, the “**product manager**”, with whom the PM should entertain a privileged relationship.

Some projects may actually be owned and/or sponsored by M&S, for example an online store or a corporate website. In that case, M&S may appoint a PM within its organization to deal with the PM in charge of project execution, to whom in some cases M&S may delegate the role of “overall PM”.

Involvement of M&S in the preliminary and preparation phases of a project

The idea leading to a project may originate within M&S or it may come from another source.
However that may be, M&S is usually involved in the **advisability study**, and is instrumental in helping the PM develop the **business case** for the project. Indeed, a product’s business model, pricing assumptions, sales projections, the budget required for promotion, etc. are usually not determined by the PM but provided by M&S.

M&S should also be involved in the **requirements specification process**, given its knowledge (in theory…) of the market, of competitors’ products and of user needs.

In particular, it is up to M&S to determine a product’s **business model**, to specify how the product will be **distributed** and, in the case of fee-based websites or online sales, what **payment methods/system** will be used.

M&S should define the various **versions of the product** needed to implement its marketing and sales strategy, for example multiple versions of a CD/DVD-ROM (“standard”, “deluxe”, demonstration versions…) or parts of a website (part free of charge, part subject to a fee).

M&S is also often responsible for providing a **graphic theme** or at least guidelines for the graphical user interface of a product.

Finally, the **deadline for the product launch** is often set by M&S (which sometimes leads to a negotiation between the PM and M&S during the project planning phase…).

In some situations, M&S may actually provide the PM with a preliminary requirements specification, which may be a simple and high-level **expression of needs**.

> *For the first version of the EHM, the Marketing department at Hachette had prepared a document that outlined the major features of the desired product, as a result of an in-depth analysis of the market and competition. The document was detailed enough to serve as a foundation for the feasibility study.*

It is the PM's responsibility to transform requirements expressed by M&S into a detailed specification. This may involve several rounds of discussion between the PM and M&S. Once the requirements specification has been finalized, it should be officially approved by M&S (preferably in writing…).

Brainstorming sessions involving M&S representatives may be useful in the process of establishing requirements. The number of participants should be limited, for example to a dozen or so, otherwise the meeting may be difficult to manage and may be less fruitful than expected. Participants should be encouraged to express themselves freely and to voice any ideas they may think of, even the boldest and craziest.

The PM will then need to sort the contributions, make sense out of them and transform them into **realistic and workable propositions**, to be reviewed and approved by those who participated in the brainstorming effort.

> *In preparation for the first redesign and redevelopment of the EHM (for its third edition, to be dated "2000"), I had devised and written an extremely detailed questionnaire, which I asked twenty or so people, including many M&S representatives, to study with great care and to fill in. The questionnaire allowed respondents to criticize the previous two versions of the EHM in terms of its editorial, functional, user-interface and technical characteristics. It also allowed them to give their feedback on a number of proposals relating to new features. The consolidated results of the survey were used as input to the requirements specification for v3 of the EHM.*
>
> *In preparation for the second redesign and redevelopment of the EHM (for its eighth edition, to be dated "2005"), I organized a series of brainstorming meetings with M&S people, selected members of the project team and representatives of the software development contractor. Involving developers was extremely beneficial since they were able to give immediate feedback concerning the feasibility of features suggested by other participants, in particular M&S people.*

Involvement of M&S in the execution phase of a project

As a major stakeholder in a project, M&S naturally has a **right of inspection** as regards its execution. Appropriate M&S representatives should therefore be invited to important meetings, in particular those at which the PM reports progress to his management, as well as those concerning key project milestones.

The **product** being created should be **shown to M&S** at various stages of its development.
The PM may be tempted to not show alpha versions to M&S because they are naturally far from complete and may not perform properly. However, since **feedback at an early stage of product development is very useful**, the PM should not hesitate to show alpha versions. In order to reassure M&S as to the quality of future versions of the product, the PM should complement the demonstrations with an appropriate commentary and justification of the current "work in progress" state of the product.

The **user interface** of a product should be submitted to M&S for review before it is "frozen". In particular, if the guidelines for the graphical interface have been provided by M&S, compliance of the interface with such guidelines must be checked by M&S.

In case of doubt or conflicting opinions concerning user-interface elements (for example the colour of a banner or the look of an icon), M&S should in principle have the last word.

When the product is in its beta stage, it is usually the right time to present it to the **sales force**.

It is sometimes useful, even necessary, to deliver **product training** to selected M&S people who are likely to give **demonstrations** of the product. The PM should provide a **demonstration scenario**, a **summary description of the product** highlighting key selling points and benefits for the users, and a "**Questions & Answers (Q&A)**" document.

When the PM believes it is the appropriate time, he should have M&S volunteers participate in "free testing" of the product. Indeed, people with a "fresh eye" often discover bugs that testers within the project team may have overlooked.

Finally, M&S needs **information from the PM in order to prepare the launch of the product**. The PM must make sure that such information is communicated to M&S in a timely fashion, as scheduled in the Communications section of the project plan.

Customer Services

The Customer Services function (aka "Support", "Assistance", "Hotline"…) is often part of the M&S department in a company's organization. However that may be, it is essential that the persons who will be in charge of **supporting the new product** resulting from a project should be given **information** and possibly **training** on the product before it is released to customers.
"Hotliners" will also require **documentation** and a "**Q&A**" on the product, as well as **the product itself**, so that they can familiarize themselves with it before it is released.
It is the PM's responsibility to make sure that Customer Services are provided with everything they need to do a good job.

It is very useful to have **hotliners** perform functional, technical and "free" **testing** of the beta version of a product. Indeed, hotliners have an excellent knowledge of how end users behave with products (for example, they are able to simulate "eccentric" usage of a product, which is liable to reveal bugs in unexpected places…). Hotliners are also familiar with product pitfalls in general, and they are usually prompt to identify any product defects.

Customer Services are also a **valuable source of information** after a product has been released, since they can relay **feedback from users** to the PM. Such information must be taken into account by the PM for new versions of the product.

It is advisable for a PM to spend time with hotliners on a regular basis, in order to become aware "in live mode" of problems encountered by users.

Cooperation between the PM and Customer Services is an essential factor in ensuring the quality of products under the PM's responsibility.

> *At Apple France, back in the years 1983-1984, the CEO required that each member of the Products team, which I was part of, should spend a whole day every month at the hotline, not as a simple observer but as an actual hotliner in direct contact with retailers and end users, which made the experience really useful and rewarding.*
>
> *At Hachette, as Director of a Products division and PM for a number of projects, I systematically involved my Customer Services colleagues in the product testing process. I also spent time on a regular basis with the hotliners, listening to and sometimes participating in conversations with end users. And I carefully analyzed and acted upon the monthly reports issued by Customer Services. The close relationship I established with Customer Services was an essential factor in ensuring a high standard of quality for "my" products.*

Product packaging

Some products need to be packaged for distribution to customers. For example, the packaging of a CD/DVD-ROM may include a plastic case or paper sleeve for the disk, a manual and a cardboard box.

M&S is generally responsible for **designing and creating** the various packaging elements. However, **text and illustrations** describing a product on its packaging are **usually provided by the PM**. If the PM is not asked for such elements, he should at least request to review the text and illustrations, in order to **make sure that product information is correct**.

The content of a product's **manual** is usually provided by the PM (or by another member of the project team), while M&S takes care of the page layout and printing.

In principle, **M&S should submit all packaging elements to the PM** for review and validation before passing them for press.
The same principle should apply to **press releases** and any other **communication material** (eg brochures, leaflets, adverts) concerning the product.

Websites

M&S may be responsible for certain elements of a website's content. Such elements may be delivered by M&S to the developer directly or through the PM. If there is a back office for the website (which is generally desirable), initial and updated content may be uploaded and published without any intervention on the part of the developer.

As guarantor of the overall quality of "his" product, **the PM should review all elements provided by M&S** and make sure that there are no semantic or syntactic mistakes.

Here is a non-exhaustive list of website content elements usually provided by M&S:

- logos and other graphic elements relating to the company's identity,
- home-page text and illustrations,
- company description and contacts,
- copyright and other legal information,
- privacy notice/policy,
- product prices,
- sales terms & conditions,
- text appearing on pages dealing with ordering and payment,
- holding pages for a provisional site.

Involvement of the PM in product promotion activities

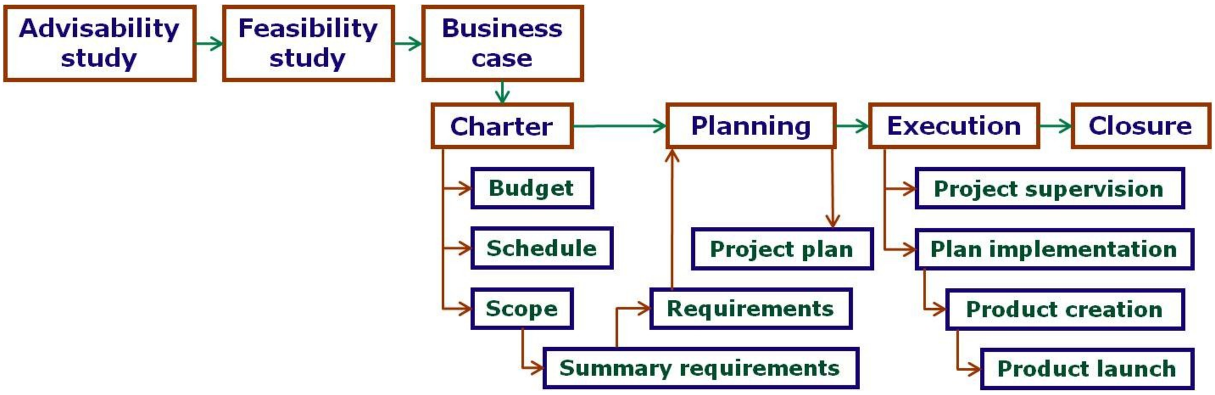
M&S is in charge of product promotion and sales, but it usually needs **help from the PM** for some of its actions and events, given the PM's in-depth knowledge of the product. The PM (or another member of the project team designated by the PM) should therefore be prepared to take part in the following M&S activities (non-exhaustive list):

- communications/advertising agency briefings,
- press conferences,
- interviews (press, radio, TV, web),
- presentations to salespeople and commercial partners (eg resellers),
- presentations to prospects and customers,
- product demonstrations at exhibitions, etc.

19) Factors of success of a project

In conclusion of this Project Management guide, I briefly address below some key factors of success of a project. This is also an opportunity to provide a recap of the various phases of a project and of areas of project management which require planning, direction/supervision, monitoring and control.

As a reminder, here is a diagram that features the main phases of a project:



The success of a project is the achievement of the objectives defined in the project charter, which generally means providing deliverables that meet requirements in terms of functionality, performance and overall quality, in compliance with budget and schedule constraints.

Needless to say, as a **prerequisite** for success the project's (and the PM's) **objectives must be "SMART"**!

- *See chapter 2: "The functions of a Project Manager".*

The **first factor** of success of a project is a **solid business case** that justifies the project and that all major stakeholders agree upon, which will guarantee support throughout the project from the project owner/sponsor and the PM's management.

- *See chapter 3: "The life cycle of a project".*

The **second factor** is a **precise charter** that clearly specifies the project's objectives with unambiguous definitions of the project's scope, budget and schedule, as well as of the PM's role and level of authority.

- *See chapter 7: "Project charter".*

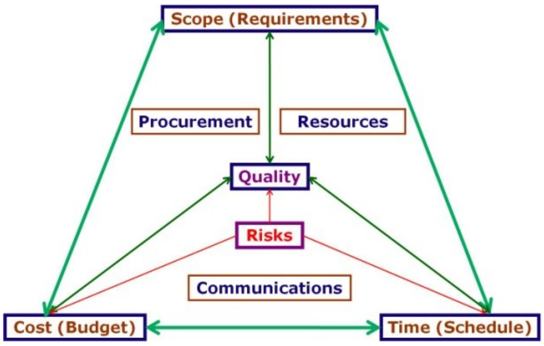
The **third factor** of success of a project is the **quality of the requirements specification**, which must be exhaustive, precise and detailed down to the appropriate level, in order to provide a clear understanding of expectations and to avoid any misinterpretation.

The requirements specification is extremely important since it is the **basis for project planning and product design**.

The requirements specification does not necessarily need to be frozen before product design is undertaken. For some projects, **a certain degree of flexibility** may actually be allowed in order to reach a "perfect match" between requirements and design. Changes to the requirements specification should however not be allowed after a certain point in the execution phase, in order to **avoid a negative impact of too many changes on cost, schedule and quality**.

- *See chapter 8: "Requirements specification".*

The **fourth factor** is **detailed and exhaustive planning** in all areas of project management, as featured in the following (now familiar) diagram:



Time spent on planning saves time in project execution. Indeed, poor and/or incomplete planning generally leads to problems during project execution, such as inadequate resources, unforeseen tasks, longer-than-expected task durations, unexpected costs.
In particular, establishing the **work breakdown structure** and determining the corresponding **sequence of tasks** are essential processes that need to be performed with extreme care and at an appropriate level of detail in order to ensure that the **estimation of task resources, durations and costs** is comprehensive and realistic.

- *See chapter 9: "Project planning".*

The **fifth factor** is the **quality and motivation of the project team**. Hiring the right people is essential. Team members must be skilled in their respective areas of expertise, they must be hard workers and have a good team spirit. They must remain motivated throughout the project, which the PM (and subproject managers, if any) must ensure.

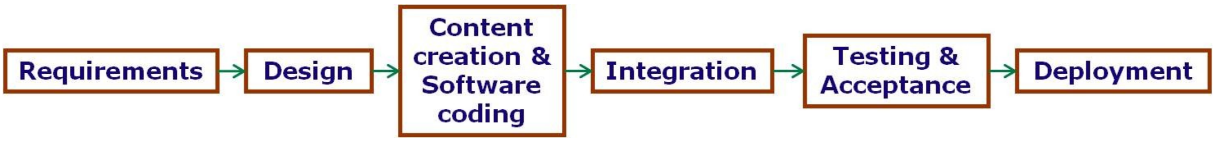
- *See chapter 10: "The project team".*

The **sixth factor** is the **quality and reliability of contractors** (and, more generally, vendors). Selecting contractors is a time-consuming process (as is hiring people) that should not be rushed. It should be done with due care and attention.

- *See chapter 11: "Contractors and contracts".*

Project success is obviously facilitated by **excellence in execution**, which is enabled by the quality of the project team and contractors, and by other factors addressed below.

The **seventh factor** is the **quality of project deliverables**, which requires rigour and excellence in execution of all tasks involved in the successive phases of product creation, as featured in the following (now familiar) diagram:



The quality of a product is dependent on **good design** (that complies with the requirements specification) and on the **quality of the product's content, user interface and software**. Of course, quality assurance implies **thorough and effective testing**.

- *See chapter 12: "Software development life cycle models and methodologies".*
- *See chapter 13: "Design specifications".*
- *See chapter 14: "Content creation".*
- *See chapter 15: "Development and developers".*
- *See chapter 16: "Testing".*

The **eighth factor** is permanent and **effective direction/supervision, monitoring and control of project execution** (ie the implementation of the project plan), which requires an **excellent organization**, a **well-oiled mechanism for information flow** and a set of **meaningful performance indicators** (which make up the "project dashboard") that are **kept up-to-date** and used to trigger **appropriate and timely action**, in particular to solve problems, to deal with issues, to settle conflicts, to arbitrate disputes, and to manage risks.

- *See chapter 17: "Project direction/supervision – Monitoring and control".*

The **ninth factor** is related to the one above; it is made explicit here because of its paramount importance: **effective communications**, which is indeed essential in order to ensure that information needed to correctly carry out project tasks is available in a timely manner to those who need it. It also enables the PM to keep abreast of what is going on and to "feed" the performance indicators. Effective communications also applies to the reporting of progress and issues to the project stakeholders.

The **tenth factor** is **close and efficient cooperation between the PM and Marketing & Sales** at various stages of the project, because the success of the project is tied to the success of the resulting product.

- *See chapter 18: "Relationship with Marketing & Sales".*

Finally (in this non-exhaustive summary), the **eleventh factor** of success is **a great Project Manager who is also a true leader**!

Indeed, the achievement of a project's objectives is highly dependent on how well it is managed. **The skills, experience and professional as well as personal qualities of the project manager, as a true leader, are key to the success of a project**.

- *See chapter 2: "The functions of a Project Manager".*
- *See chapter 17: "Project direction/supervision – Monitoring and control".*

Here are a few links to interesting **articles relating to the subject of this chapter**:

>> www.projectsmart.com/articles/eight-key-factors-to-ensuring-project-success.html

>> www.projectsmart.co.uk/six-ways-to-give-proper-project-leadership.html

>> https://www.pmhut.com/the-abcs-of-project-management-for-project-managers

---

As a **final note**, I want to offer the following personal opinion.

- **The job of Project Manager is probably one of the most challenging and intellectually rewarding**.

☺