Project Management Handbook

Version 1.1 - July 2006

Wouter Baars

Recommendations:
Henk Harmsen
Rutger Kramer
Laurents Sesink
Joris van Zundert

DANS – Data Archiving and Networked Services

The Hague – 2006

DANS – Data Archiving and Networked Services
PO Box 93067
2509 AB The Hague

T +31 (0)70-3494450
F +31 (0)70-3494451
info@dans.knaw.nl
www.dans.knaw.nl

ISBN 90 6984 496 6

Table of Contents

Foreword

Anyone who has ever worked on a project will agree that making a project succeed is no simple task. The difficulties manifest themselves in (extreme) delays, (extreme) budget over-runs, inadequate results, dissatisfied customers, high stress among the project team and other undesirable outcomes. What is the cause of all of these problems?

Projects are characterised by four features: a group of people, a goal, limited time and money and a certain level of uncertainty regarding whether the goals will be achieved. Project managers are involved with all of these aspects. Supervising and directing a project is thus anything but an easy task.

Projects are becoming increasingly common. Project-based working methods have also found their way into non-profit organisations, including DANS.[1] The rules of the game for projects in non-profit organisations differ from those in commercial organisations. Political factors play a particularly important role in non-profit organisations. This makes it even more difficult for projects to succeed, compared to projects in which commercial aspects play a part. Project leaders should be aware of this and be able to play the game of politics.

After several years of experience with IT projects, the authors of this handbook have become even more keenly aware of how IT projects differ from 'regular' projects. Most importantly, projects are more dynamic, and that has both advantages and disadvantages. We have established that IT projects require an approach that differs – at least partly - from the approaches that are appropriate for construction, re-organisations or other types of projects.

This handbook is intended for projects that are conducted by DANS. The first section describes a working method that can be followed for 'traditional' projects. The second section describes the working method for IT projects, particularly those that involve software development. This handbook presents a practical model that will allow project members, project leaders, project managers, general managers, programme managers, customers and project partners to play their roles within DANS better.

It is impossible to learn all there is to know about the field of project management. Theoretical development and practical experience are continually producing new insights. This handbook is therefore incomplete, and it will grow along with new developments in the area of project management. To make this possible, we have chosen to publish the text under a creative-commons license. This means that anyone is free to use, copy or change the text.[2] Most importantly, it means that anyone who feels that the text is in need of additions or improvement should not hesitate to do just that!

Henk Harmsen
Deputy Director
DANS
The Hague, May 2006

[1] Data Archiving and Network Services (DANS) is a joint initiative of the Royal Netherlands Academy of Arts and Sciences (KNAW) and the Netherlands Organisation for Scientific Research (NWO), with the goal of improving the scientific data structure in the Netherlands.

[2] For the exact terms of the license, please refer to Appendix 4.

Introduction

This project management handbook is intended for anyone who is involved in or will be involved in projects that take place within or are conducted in association with DANS. The text, however, has been prepared in such a way that it can be used by other organisations, particularly those in the non-profit sector, that use project-based working methods.

The book is comprised of several sections. The first section (Chapters 1 through 4) provides an overview of project management. These chapters address the theory of the waterfall method, which is applicable to most projects. The second section of this book (beginning with Chapter 5), addresses 'cyclical' forms of project management, which are more appropriate to IT-related projects. These methods are particularly well suited for software development and other creative IT projects. The penultimate chapter addresses the working methods of DANS. This method is a combination of elements from both the waterfall and the cyclical methods. The last chapter of this handbook discusses how organisations can manage the dynamics of carrying out several projects at once. The most important difficulties are addressed, along with strategies for dealing with these problems.

'This document includes a number of standard documents that can be used for directing projects, as well as a number of references to open-source project instruments developed by third parties. A literature list is included at the end of this book for those who wish to delve more deeply into the broad field of project management.

1. The six phases of project management

This chapter provides a sketch of the traditional method of project management. The model that is discussed here forms the basis for all methods of project management. Later chapters go into more depth regarding a model that is particularly appropriate for IT-related projects.

Dividing a project into phases makes it possible to lead it in the best possible direction. Through this organisation into phases, the total work load of a project is divided into smaller components, thus making it easier to monitor. The following paragraphs describe a phasing model that has been useful in practice. It includes six phases:

1. Initiation phase
2. Definition phase
3. Design phase
4. Development phase
5. Implementation phase
6. Follow-up phase

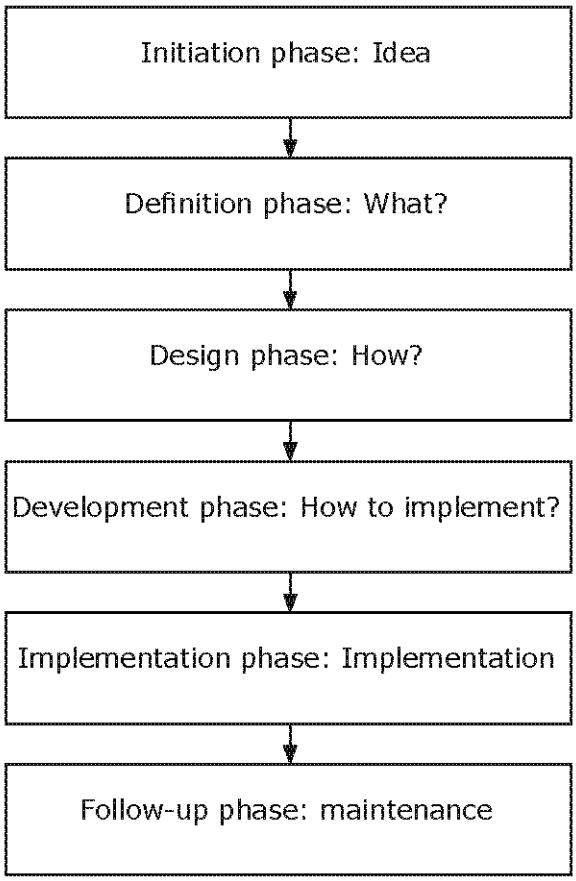


Figure 1: Project management in six phases, with the central theme of each phase

Initiation phase

The initiation phase is the beginning of the project. In this phase, the idea for the project is explored and elaborated. The goal of this phase is to examine the feasibility of the project. In addition, decisions are made concerning who is to carry out the project, which party (or parties) will be involved and whether the project has an adequate base of support among those who are involved.

In this phase, the current or prospective project leader writes a proposal, which contains a description of the above-mentioned matters. Examples of this type of project proposal include business plans and grant applications. The prospective sponsors of the project evaluate the proposal and, upon approval, provide the necessary financing. The project officially begins at the time of approval. Questions to be answered in the initiation phase include the following:

- Why this project?
- Is it feasible?
- Who are possible partners in this project?
- What should the results be?
- What are the boundaries of this project (what is outside the scope of the project)?

The ability to say 'no' is an important quality in a project leader. Projects tend to expand once people have become excited about them. The underlying thought is, 'While we're at it, we might as well …' Projects to which people keep adding objectives and projects that keep expanding are nearly certain to go off schedule, and they are unlikely to achieve their original goals.

In the initiation phase, the project partners enter a (temporary) relationship with each other. To prevent the development of false expectations concerning the results of the project, it makes sense to explicitly agree on the type of project that is being started:

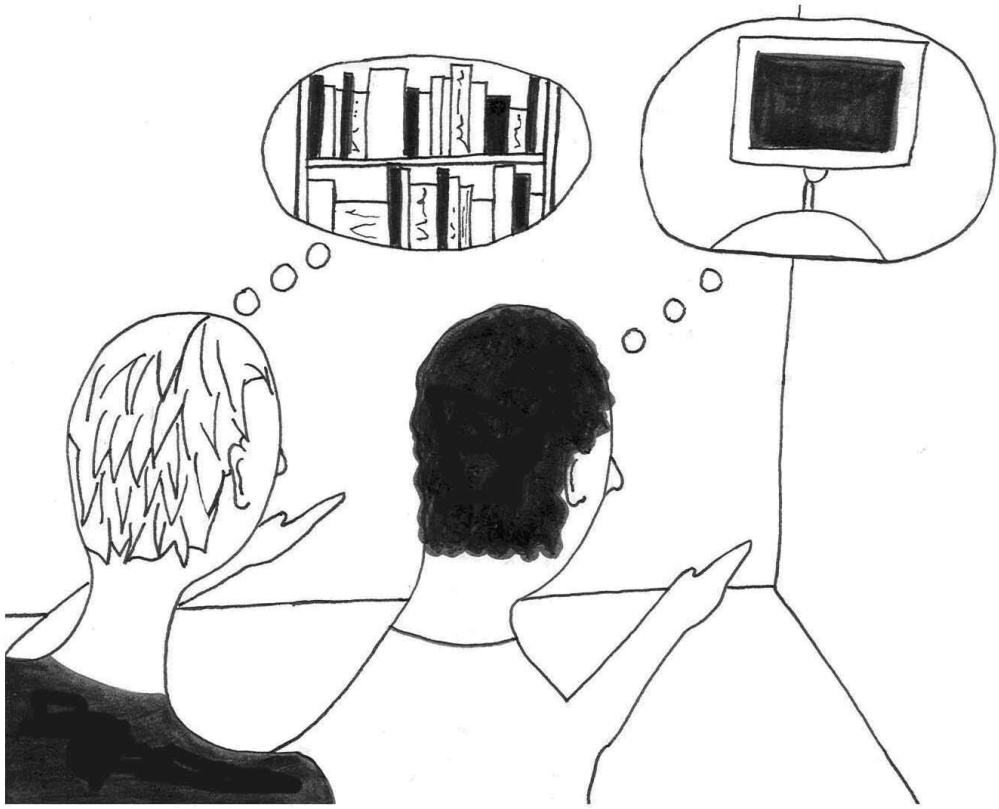
- a research and development project;
- a project that will deliver a prototype or 'proof of concept';
- a project that will deliver a working product.

The choice for a particular type of project largely determines its results. For example, a research and development project delivers a report that examines the technological feasibility of an application. A project in which a prototype is developed delivers all of the functionalities of an application, but they need not be suitable for use in a particular context (e.g. by hundreds of users). A project that delivers a working product must also consider matters of maintenance, instructions and the operational management of the application.

Many misunderstandings and conflicts arise because the parties that are involved in a project are not clear on these matters. Customers may expect a working product, while the members of the project team think they are developing a prototype. A sponsor may think that the project will produce a working piece of software, while the members of the project team must first examine whether the idea itself is technically feasible.

Definition phase

After the project plan (which was developed in the initiation phase) has been approved, the project enters the second phase: the definition phase. In this phase, the requirements that are associated with a project result are specified as clearly as possible. This involves identifying the expectations that all of the involved parties have with regard to the project result. How many files are to be archived? Should the metadata conform to the Data Documentation Initiative format, or will the Dublin Core (DC) format suffice? May files be deposited in their original format, or will only those that conform to the 'Preferred Standards' be accepted? Must the depositor of a dataset ensure that it has been processed adequately in the archive, or is this the responsibility of the archivist? Which guarantees will be made on the results of the project? The list of questions goes on and on.



'After the brain-storming session, all of the members of the "New Archive" project team were in agreement about the desired outcome'.

Figure 2: Expectations of a project (Illustration: Rachèl Harmsen)

It is important to identify the requirements as early in the process as possible. Wijnen (2004) distinguishes several categories of project requirements that can serve as a memory aid:

- Preconditions
- Functional requirements
- Operational requirements
- Design limitations

Preconditions form the context within which the project must be conducted. Examples include legislation, working-condition regulations and approval requirements. These requirements cannot be influenced from within the project. *Functional requirements* are requirements that have to do with the quality of the project result (e.g. how energy-efficient must an automobile be or how many rooms must a new building have?). *Operational requirements* involve the use of the project result. For example, after a software project has been realised, the number of malfunctions that occur must be reduced by ninety per cent. Finally, *design limitations* are requirements that involve the actual realisation of the project. For example, the project cannot involve the use of toxic materials or international partners for whom it is unclear whether they use child labour.

During the definition phase of a project that involved developing a web application for a consortium of large organisations, no agreements were made concerning the browser that would be supported by the application. The consortium assumed that it would be Microsoft Explorer, because it was the browser that 'everyone' used. The programmers created the application in Firefox, because they worked with the browser themselves and because it had a number of functions that were particularly useful during the development. Because most of the websites that are made for Firefox also look good in Explorer, the difference was initially not noticeable. Near the end of the project, however, the customer began to complain that the website 'didn't look good'. The programmers, who had been opening the site in Firefox, did not understand the complaint.

When the problem of the two browsers became clear, the programmers reacted defensively, 'Can't they just install Firefox? After all, it *is* free'. The organisations, however, were bound to the bureaucratic-minded system administrators who, for some possibly justified reason, refused to install Firefox in addition to Explorer. Even if they had wanted to install it, it would have involved a lengthy process, and there would have been extra costs for the time that the system administrators would have to spend on the task. It was ultimately decided that the application would have to be made suitable for Explorer. That involved considerable extra work, whereby the project ran even more behind schedule than it already had, and it was necessary to negotiate the extra costs. It was later discovered that the various organisations were working with different versions of Microsoft Explorer.

It is very important that all parties that are involved in the project are able to collaborate during the definition phase, particularly the end users who will be using the project result. The fact that end users are often not the ones that order the project perhaps explains why they are often ignored. The client, who pays for the project, is indeed invited to collaborate on the requirements during the definition phase. Nonetheless, the project result benefits when its future users are also invited. As a point of departure, it is helpful to make a habit of organising meetings with all concerned parties during the definition phase of a project.

During the development of an educational video game, the users (young people) were involved in the project only at a later stage. When the game was nearly completed, a group of young people was asked to test the game. Their initial assessments appeared mild and friendly. When pressed, however, they admitted that they had actually found the game 'extremely boring' and that they would 'certainly not play it themselves'. Had these young people been involved in the project earlier, the game would probably have been a success. As it stands, the game remains nearly unused on an Internet website.

The result of the definition phase is a list of requirements from the various parties who are involved in the project. Every requirement obviously has a reverse side. The more elaborate the project becomes, the more time and money it will cost. In addition, some requirements may conflict with others. New copy machines are supposed to have less environmental impact; they must also meet requirements for fire safety. The fire-safety regulations require the use of flame-retardant materials, which are less environmentally friendly. As this illustration shows, some requirements must be negotiated.

Ultimately, a list of definitive requirements is developed and presented for the approval of the project's decision-makers. Once the list has been approved, the design phase can begin. At the close of the definition phase, most of the agreements between the customer and the project team have been established. The list of requirements specifies the guidelines that the project must adhere to. The project team is evaluated according to this list. After the definition phase, therefore, the customer can add no new requirements.

A part of a new exhibit in a museum was comprised of a computer installation, the creation of which had been project-based. Because there had been no definition phase in the project, no clear agreements between the museum and those responsible for building the installation had been made. When the computer for the installation broke down halfway through the exhibit, the museum assumed that it would be covered by the project's guarantee. The project team had a different opinion. Negotiations between the directors were necessary in order to arrive at an appropriate solution.

Design phase

The list of requirements that is developed in the definition phase can be used to make design choices. In the design phase, one or more designs are developed, with which the project result can apparently be achieved. Depending on the subject of the project, the products of the design phase can include dioramas, sketches, flow charts, site trees, HTML screen designs, prototypes, photo impressions and UML schemas. The project supervisors use these designs to choose the definitive design that will be produced in the project. This is followed by the development phase. As in the definition phase, once the design has been chosen, it cannot be changed in a later stage of the project.

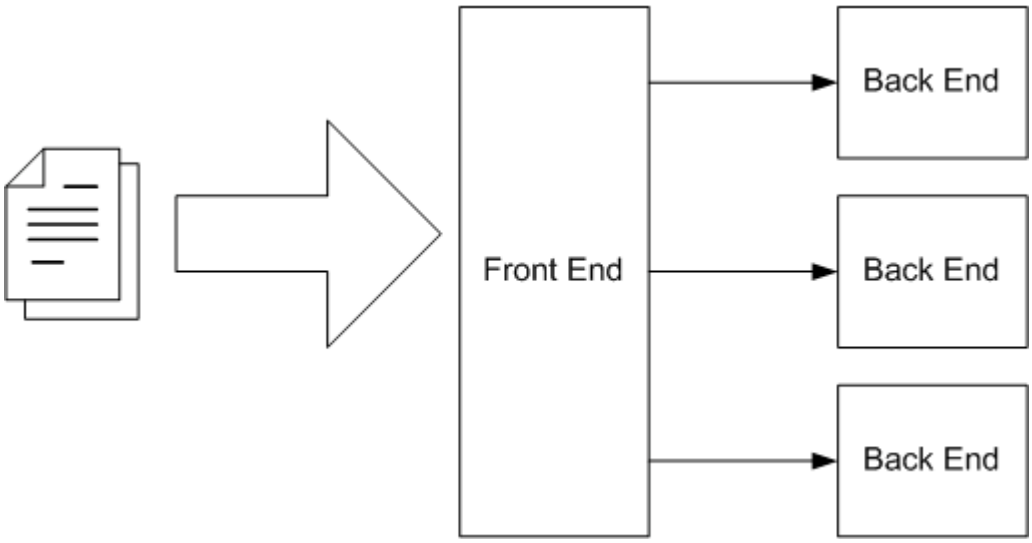


Figure 3: Example: Global design for the DANS Architecture Archive

In a young, very informal company, the design department was run by an artist. The term 'design department' was not accurate in this case; it was more a group of designers who were working together. In addition, everyone was much too busy, including the head of the department.

One project involved producing a number of designs, which were quite important to the success of the project. A young designer on the project team created the designs. Although the head of the design department had ultimate responsibility for the designs, he never attended the meetings of the project team when the designs were to be discussed. The project leader always invited him, and sent him e-mails containing his young colleague's sketches, but the e-mails remained unanswered. The project leader and the young designer erroneously assumed that the department head had approved the designs. The implementation phase began. When the project was nearly finished, the result was presented to the department head, who became furious and demanded that it be completely redone. The budget, however, was almost exhausted.

Development phase

During the development phase, everything that will be needed to implement the project is arranged. Potential suppliers or subcontractors are brought in, a schedule is made, materials and tools are ordered, instructions are given to the personnel and so forth. The development phase is complete when implementation is ready to start. All matters must be clear for the parties that will carry out the implementation.

In some projects, particularly smaller ones, a formal development phase is probably not necessary. The important point is that it must be clear what must be done in the implementation phase, by whom and when.

Implementation phase

The project takes shape during the implementation phase. This phase involves the construction of the actual project result. Programmers are occupied with encoding, designers are involved in developing graphic material, contractors are building, the actual reorganisation takes place. It is during this phase that the project becomes visible to outsiders, to whom it may appear that the project has just begun. The

implementation phase is the 'doing' phase, and it is important to maintain the momentum.

In one project, it had escaped the project team's attention that one of the most important team members was expecting to become a father at any moment and would thereafter be completely unavailable for about a month. When the time came, an external specialist was brought in to take over his work, in order to keep the team from grinding to a halt. Although the team was able to proceed, the external expertise put a considerable dent in the budget.

At the end of the implementation phase, the result is evaluated according to the list of requirements that was created in the definition phase. It is also evaluated according to the designs. For example, tests may be conducted to determine whether the web application does indeed support Explorer 5 and Firefox 1.0 and higher. It may be determined whether the trim on the building has been made according to the agreement, or whether the materials that were used were indeed those that had been specified in the definition phase. This phase is complete when all of the requirements have been met and when the result corresponds to the design.

Those who are involved in a project should keep in mind that it is hardly ever possible to achieve a project result that precisely meets all of the requirements that were originally specified in the definition phase. Unexpected events or advancing insight sometimes require a project team to deviate from the original list of requirements or other design documents during the implementation of the project. This is a potential source of conflict, particularly if an external customer has ordered the project result. In such cases, the customer can appeal to the agreements that were made during the definition phase.

As a rule, the requirements cannot be changed after the end of the definition phase. This also applies to designs: the design may not be changed after the design phase has been completed. Should this nonetheless be necessary (which does sometimes occur), the project leader should ensure that the changes are discussed with those involved (particularly the decision-makers or customers) as soon as possible. It is also important that the changes that have been chosen are well documented, in order to prevent later misunderstandings. More information about the documentation of the project follows later in this handbook.

Follow-up phase

Although it is extremely important, the follow-up phase is often neglected. During this phase, everything is arranged that is necessary to bring the project to a successful completion. Examples of activities in the follow-up phase include writing handbooks, providing instruction and training for users, setting up a help desk, maintaining the result, evaluating the project itself, writing the project report, holding a party to celebrate the result that has been achieved, transferring to the directors and dismantling the project team.

The central question in the follow-up phase concerns when and where the project ends. Project leaders often joke among themselves that the first ninety per cent of a project proceeds quickly and that the final ten per cent can take years. The boundaries of the project should be considered in the beginning of a project, so that the project can be closed in the follow-up phase, once it has reached these boundaries.

It is sometimes unclear for those concerned whether the project result is to be a prototype or a working product. This is particularly common in innovative projects in which the outcome is not certain. Customers may expect to receive a product, while the project team assumes that it is building a prototype. Such

situations are particularly likely to manifest themselves in the follow-up phase. Consider the case of a software project to test a very new concept.

There was some anxiety concerning whether any results would be produced at all. The project eventually produced good results. The team delivered a piece of software that worked well, at least within the testing context. The customer, who did not know much about IT, thought that he had received a working product. After all, it had worked on his office computer. The software did indeed work, but when it was installed on the computers of fifty employees, the prototype began to have problems, and it was sometimes instable.

Although the programmers would have been able to repair the software, they had no time, as they were already involved in the next project. Furthermore, they had no interest in patching up something that they considered a trial piece. Several months later, when Microsoft released its Service Pack 2 for Windows, the software completely stopped functioning. The customer was angry that the 'product' once again did not work. Because the customer was important, the project leader tried to persuade the programmers to make a few repairs. The programmers were resistant, however, as repairing the bugs would cause too much disruption in their new project. Furthermore, they perceived the software as a prototype. Making it suitable for large-scale use would require changing the entire architectural structure. They wondered if the stream of complaints from the customer would ever stop.

The motto, 'Think before you act' is at the heart of the six-phase model. Each phase has its own work package. Each work package has its own aspects that should be the focus of concentration. It is therefore unnecessary to continue discussing what is to be made during the implementation phase. If all has gone well, this was already determined in the definition phase and the design phase. For a more detailed description of the six-phase model and the task packets for each phase, see Wijnen (2004) and Kor (2002).

2. Managing a project

Adopting the six phases creates clarity in a project, thereby making it easier to administer. What exactly does managing a project entail?

First, project leaders and project teams are involved with the following components:

1. Team
A project team is comprised of a group of people who will realise the project result. The group is often comprised of people who have various backgrounds, each of whom contributes knowledge and skills.

2. Goal
A product result (or goal) is desired. After a project has been completed, something has been realised. A new piece of software has been written, a re-organisation has been carried out or a bridge has been built. The project goal is sometimes vague or less firmly established. In many projects, it is necessary to adapt the goal as the project proceeds.

3. Limited resources
The amount of time and money that is available for completing a project is always limited. No project is completely free of time pressure.

4. Uncertainty (risk)
One characteristic feature of projects is that their success is never guaranteed beforehand. Even if the desired goal is already being reached, it is uncertain whether it will be achieved within the available budget or within the proposed time. It is not unusual for a project to take three times as long and to cost twice as much as originally estimated. It is also not unusual for only thirty per cent of the original project team members to be working on the project upon its completion.
Although project managers must attend to many matters, they actually direct projects along only five parameters:

- Time
- Money
- Quality
- Organisation
- Information

These five parameters, which are often known as the 'control factors', are described further below. The control factors appear in project plans, progress monitoring and project reporting.

Time

The time factor manifests itself in a project in the form of deadlines for tasks and the amount of time that these tasks may take. Managing time involves ensuring that tasks are completed on time.

Time in project plans:

- Determine which activities should take place in which phase.
- Estimate how long each activity will take
- Determine the order in which activities should be completed.
- Allocate people and materials.
- Allocate activities over time.
- Determine the (most important) deadlines.

Time in progress monitoring:

- Monitor progress.
- Monitor deadlines.
- Adjust schedules.

Time in project reporting:

- Report on the actual timeline.
- Analyse and explain why some tasks proceeded much more quickly or much more slowly than expected.

Time schedules are based on a work-breakdown structure (WBS). A WBS is a decomposition of the tasks that must be completed in order to achieve the project result. Developing a time schedule requires knowing the amount of time that is needed for each task, who will complete each task and when. One frequently used tool for planning time is the bar chart or Gantt chart (see (1) Material purchasing (2) Material testing (3) Compile testing report (4) Edit report (5) Information days Figure 5 A variety of software packages is available for making and maintaining bar charts (see Appendix 3).

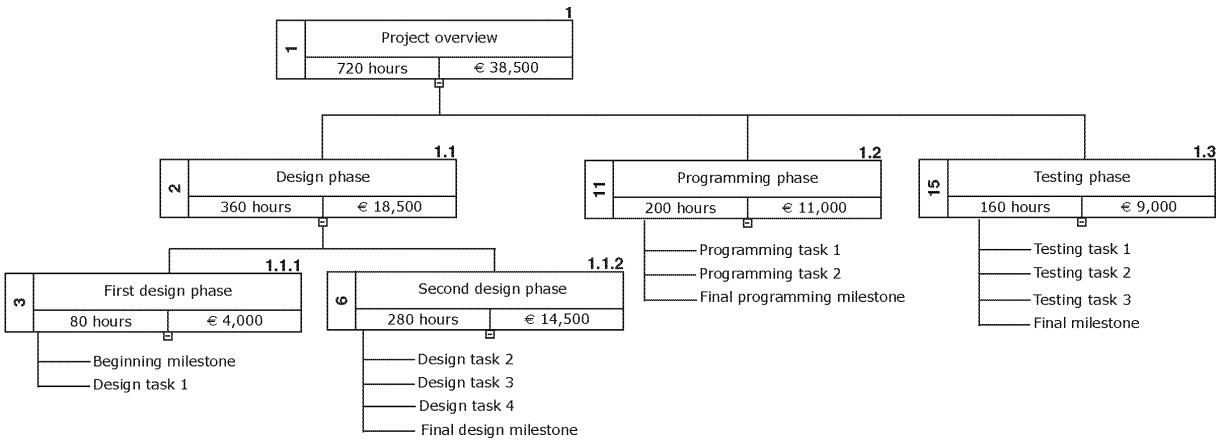


Figure 4: A (portion of a) WBS of a project

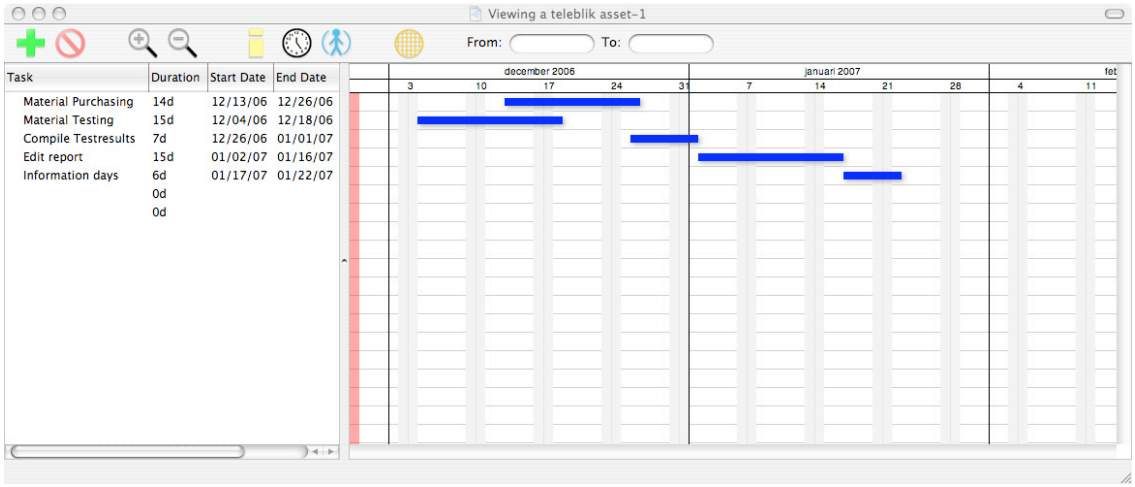


Figure 5: Gantt chart or bar chart, which is commonly used for time planning.

A rapidly growing organisation was continually taking on more projects. As the company continued to become busier – its products were in great demand – the personnel began to feel pressured to work in a frenzy to complete all of the work that needed to be done. The personnel wanted more people to be hired. Because of the cost, management was hesitant to do so, and they pressured the existing personnel to work harder. How much work could the team actually handle? This question apparently had no good answer, as the organisation had no time-registration system.

When a new project was started, an estimate was made of the number of hours that was thought necessary, but no one ever checked during or after the project to determine whether this number of hours was actually needed. Project leaders were nonetheless urged to keep their projects under control. The project leaders protested that, without time records, they had no oversight over the projects. After all, because they had no insight into the number of hours that were used to carry out the tasks of a project, and there was absolutely no chance of adjustment.

One project leader decided to register hours with his team. The registration showed that the project ultimately needed four times as many hours as had been originally estimated. After reprimanding the project leader for allowing the project to get so far out of hand, the management decided to introduce a time-registration system.

After several months, a number of bottlenecks became apparent. It was revealed that nearly all of the projects had been budgeted too narrowly. In practice, personnel who had been assigned to work on a project for one hundred hours often proved to need three times as many hours. This transparency was accompanied by new dilemmas. One the one hand, there were indeed too few personnel to carry out the projects well. Additional personnel were needed. The costs of sufficient personnel were considerable. On the other hand, the projects had apparently been sold far too cheaply (for too few hours) to customers. The management was afraid that they would not receive any more orders if they began to charge more hours in their estimates.

Money

The money factor manifests itself in the project budget. The management of money within a project involves ensuring that the costs remain within the budget. Given that the majority of the costs in most projects are comprised of labour costs, the factors of money and time (the number of labour hours) are closely intertwined.

Money in project plans:
- Determine the fees of the team members.
- Estimate the hours for the team members.
- Assign budgets to team members for specific tasks.
- Determine costs for material and tools.

Money in progress monitoring
- Monitor cash flow.
- Negotiate with suppliers.
- Determine whether the original cost estimates are still accurate.
- Adjust budgets.
- Negotiate with customer and/or client concerning budget adjustments.

Money in project reporting:
- Compile financial reports and statements.
- Analyse definitive financial report.

Quality

The project result must fulfil a number of quality requirements. This also applies to the various intermediate products of the project. When managing a project, it is particularly important for quality requirements to be determined, agreed upon and recorded in writing during the definition phase. These requirements should never remain implicit. A clear list of requirements can be checked at the end of the implementation phase. This can allow the project team to prove that they have carried out the project according to specifications. Additional quality requirements may be specified for various tasks within the project. For example, a particular task can be carried out only by certified personnel.

Quality in project plans:
- Establish the desired quality of the project result and the intermediate products (this takes place primarily in the definition phase).
- Establish the desired quality of the carrying out of the various activities in the project.

Quality in progress monitoring:
- Test the (intermediate) results.
- Address any quality problems.

Quality in the project reporting:
- Confirm that the desired quality has been attained.
- Address any complaints (particularly in the follow-up phase).

Perfectionism impedes project management. A pragmatic attitude toward the quality levels of a project can be expressed as ‘Good enough is good’. Projects that

strive to achieve the highest possible level of quality are often at great risk of never being completed.

Organisation

Within a project, the team must be managed. In the narrowest sense, team management involves determining who will do what from the list of activities. In broader terms, it also involves all of the soft skills (e.g. motivational techniques, communication skills, leadership styles) that are needed to achieve a goal with a group of people. Regardless of their importance, these soft skills exceed the scope of this handbook.

Organisation in project plans:

- Assemble the team.
- Assign authority.
- Assign tasks to team members.
- Make agreements concerning the availability of people with other (project) managers and higher management.

Organisation in progress monitoring:

- Direct the team.
- Monitor human aspects (soft skills).
- Mediate between the parties who are involved in the project.

Information

The information factor concerns how, by whom and on which basis decisions can be taken. Who may decide about which matters in the project? Is it the project leader, the client or a substantive expert within the team? What will be archived and by whom? Will tools (e.g. project website, issue tracker, e-mail notification, joint agenda) be used? These and other informational questions must be answered before a project can be started. Organisations that regularly work with projects have a number of tools (e.g. Word templates) on hand for handling information within a project.

Information in project plans:

- Which information must be provided to whom and in which form?
- Which information will be recorded, distributed and archived?
- Which information tools will be used?

Information in progress monitoring:

- Arrange for periodic consultation.
- Ensure that the right information is provided to the right person.
- Determine whether agreements have been met.

Information in project reporting:

- Write the project report.

Appendices 6 through 9 of this handbook provide a number of samples of information forms that can be used for exchanging information exchange within a project:

- Issue list
- Action-and-decision list

- Risk log
- Meeting report

The *issue list* contains all of the points that must be discussed. This list must be discussed regularly. For keeping track of progress and registering decisions that have been taken, a model for an *action and decision list* has been included. A *risk log* has been included to help document risks that are identified during a project. These risks must then be discussed in the next meeting of the project team and, where necessary, eliminated. Finally, a standard *meeting report* has been included as an example of how to compiled and archive this type of report. Appendix 3 contains an overview of helpful tools by third parties.

One important aspect of securing the information concerning a project is that all decisions should be reproducible. Decisions are often taken orally and not archived. Regardless of how clear such decisions may seem at the time for both parties, they must eventually appear in writing. If this is not possible, the undocumented decision can become a source of misunderstanding or even conflict.

Many projects are delayed by various interventions from outside (e.g. 'this is even more important', 'this is better politically', 'the customer wants us to work on something else first'). Keeping a personal log for recording this type of intervention can help project workers identify the cause of project delays.

3. Project reporting

Crucial decisions must be taken at five points during a project; these decision points correspond to the end of each project phase, and they call for recording the project's current status and writing an intermediate report. They also provide the opportunity to reconsider the project phases that are yet to come.

At these decision points, project leaders should consult with their clients regarding decisions about the project and adjust the control factors, if necessary. For example, if many new and unexpected requirements have emerged during the definition phase that could increase the costs considerably, it is useless to proceed with the original budget.

The decision points at the end of the phases are often 'go/no-go moments'; they call for decisions regarding whether to proceed with the project or whether it should be discontinued.

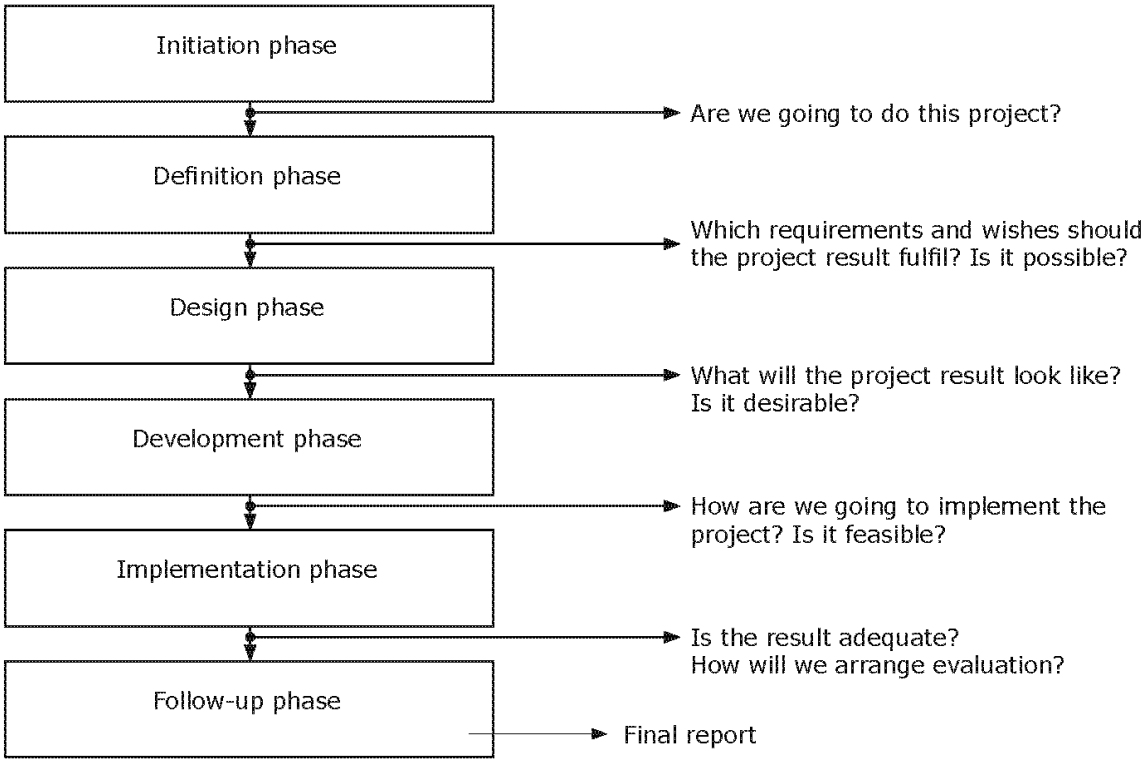


Figure 6: Five important decision points in a project

The following situation often occurs in organisations that do not work according to project phases: a project plan is initially written, in which the control factors are described. A timeline (Time) is specified, and a budget (Money) is prepared, a team is formed (Organisation), a goal is described (Quality) and the tools for information services surrounding the project are determined (Information). During a project, the project leader continues to make sure that the project remains somewhat within the total budget and the timeline, but makes no real adjustments. Near the end of a project, the project proves to cost more or to take longer than originally expected. The project is then scaled back to avoid further cost over-runs or delays. Unfortunately, the project result suffers.

Had the project leader in such a case worked with the six-phase model, the team would probably have already concluded in the design phase or perhaps even in the definition phase that the original timeline and budget were insufficient. If the project leader had made adjustments at that time, a simpler design could have been chosen that would have been less expensive and time-consuming to implement. Alternatively, more time, money or both could have been requested from the client. At any rate, the status of the project would have been clear months earlier, and it would have been possible to steer the project in a meaningful way.

Uncertainty in project plans

Projects involve uncertainty. At the beginning of a project, the exact amount of time that will be needed is not known, nor is the precise amount that the project will eventually cost. For some projects, it is even uncertain whether the intended goal will be reached at all. In a world of fast-paced change, the foundations of a project have sometimes already changed before the project is completed. This sometimes occurs because of technological developments or developments in the market or political arena.

When preparing project plans, project leaders can only estimate the control factors (i.e. time, money, team, quality goals and necessary information) of the project. As the project proceeds, more knowledge emerges about the project itself. In the initiation phase, only an idea exists. In the definition phase, the idea is refined according to requirements. In the design phase, possible designs are examined and developed, providing even more clarity. In the development phase, it becomes clear how the design should be realised. In the implementation phase, the actual project result is built, and in the follow-up phase, all of the loose ends are tied together.

Clarity increases as a project progresses. It is therefore pointless to make a detailed budget for the follow-up phase (which will take place later) during the initiation phase. At this stage, it is still possible for the project to proceed in any of a number of possible directions. The idea has yet to be elaborated. The exact form of the follow-up phase is probably also known only in the broadest terms. This is too little information upon which to base a realistic, detailed estimate for the follow-up phase. A broad outline of a budget is the most that can be expected at this stage.

Project plans therefore work as follows: a global budget is made for the entire project, along with a concrete budget for the next subsequent phase. For example, if a project team is preparing to enter the implementation phase (after the development phase), they are well aware of what must happen. At that point, it is possible to make a detailed budget for the implementation phase.

The global budget estimates for the total project must be adjusted after each phase. After each phase, there is more knowledge and decisions have been taken that allow the global budget to be completed in more detail. In this way, estimates of the total costs of the project become increasingly accurate after each phase.

Making a global budget for the entire project and a concrete budget for the next phase is important, and not only for the control factor of money. It is important to work from global to concrete for the other factors as well.

The process of making budget estimates can be summarised as follows:

- Budgeting should occur before each phase.
- Make or adjust the global budget for the entire project.
- Make a concrete budget for the next phase.
- All control factors should be reconsidered and re-estimated for each new phase.

Budgeting in this way (particularly with regard to time and money) is a realistic manner of coping with uncertainty, which is greater at the beginning of the project than it is at the end. It creates a problem, however, for organisations that are financed by government subsidies, social foundations or both. This is particularly true for organisations that conduct innovative, and thus uncertain, projects.

Most foundations and grant makers require a project proposal that includes a complete and firmly established budget before they will release funds for a project. An organisation that seeks financing for a project must therefore develop a complete, concrete budget at a very early stage. In the beginning, however, the project is still in the conceptual phase, and it is thus impossible to make a realistic cost estimate or timeline. Only after the design phase, when the idea has been elaborated and a design has been chosen, is there sufficient information to say how much the project will cost and how much time its implementation will take. This stage does not occur until several months after the grant application must be submitted.

One result of the way in which grant makers and foundations tend to work is that many organisations request amounts that are based on rough estimates of the project costs. Project activities are subsequently fitted to the budget that has been made available. This puts the project team in a tight position from the start, even though the most flexibility is needed in the early stages.

The process of elaborating concepts during the definition and design phases, therefore, often reveals that the timeline that was proposed in the grant application is not feasible. The budget may also prove inadequate, including too much for some items and not enough for others. Any additional requirements from the grant maker (e.g. no item may deviate more than five per cent) place the project team under immense pressure. Matters must be implemented in too little time and within a budget that is too tight. This situation often leads to considerable shuffling among the various items in the budget. Considerable text and analysis is then necessary in the project statement to explain why the desired result was not achieved.

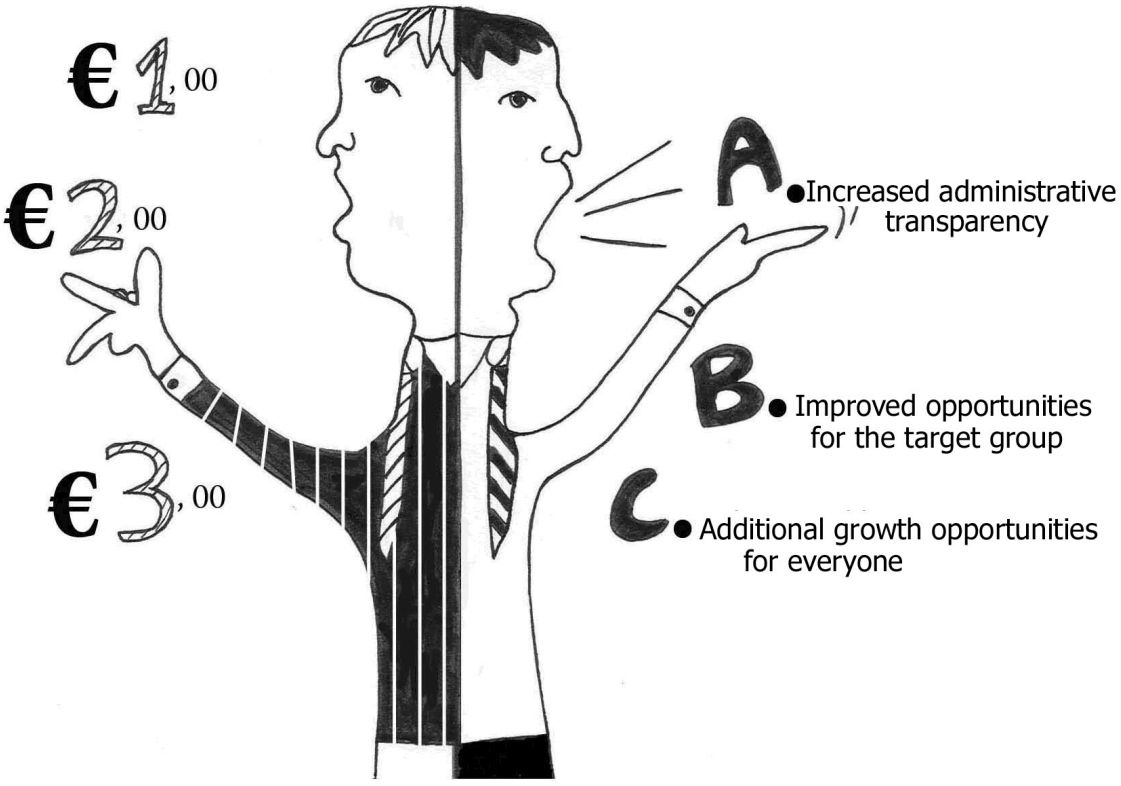
The situation would improve if grant makers were to couple their financing onto the various phases instead of providing funds at one time in advance. The initial financing would then be intended for the definition and the design phases. The requirements would be investigated and a number of alternative designs would be prepared within this limited budget. A subsequent application based on these designs would then be submitted for implementation and follow-up. This would allow projects to avoid unnecessary pressure. An additional advantage would be that the expectations of the involved parties would be more realistic, saving time, money and disappointment.

4. The sales representative and the politician

Project leaders should consider the environment within which their projects will take place. In other words, they should consider the ways in which decisions are taken within and about the project. A project may be located in one of two worlds: the world of the sales representative and the world of the politician.

The world of the sales representative revolves around profit maximisation, and stability is very important. Actions are based on mutual trust and are subject to the motto of, 'a deal is a deal'. Relationships among sales representatives are important, and the behaviour that they exhibit is genuine. Power is decentralised.

In the world of the politician, the majority is important for getting things done. Loyalty to the group is thus important, even if a politician's opinion differs from that of the group on a number of points. Because the majority seldom consists of a single group, temporary coalitions are often necessary, sometimes with opponents or even enemies. Decisions emerge from a particular view of the world. In the world of the politician, references to certain facts are necessary to maintain good order; the end justifies the means. Power is centralised.



The sales representative and the politician

Figure 7: The sales representative and the politician (Illustration: Rachèl Harmsen)

Most people intuitively feel more attracted to the first of these two worlds; the second brings many negative associations to mind 'We don't play politics here' is a frequently heard remark in organisations, even if it is not true. Even though most people find the world of the sales representative the more attractive of the two, it has an important disadvantage. Decision-making according to profit maximisation works only for decisions in which clear cash flows are available. Decisions that involve such dilemmas or issues as investing more in education, the environment, health care, highways, research, defence or nuclear energy cannot be expressed as an unambiguous balance between profit and loss. The political model is the only possible model for such decisions. It is therefore necessary to play the political game.

By definition, social and subsidised organisations exist within the world of the politician. The financing of these organisations and their projects is completely or largely dependent upon the political will to support the organisation. The effectiveness of social organisations is not easily expressed in terms of cash flows. This is also true of the results of projects that are carried out by social organisations.

A young engineer once had to carry out an ambitious wind-energy project in a municipality somewhere in the country. Through an ingenious savings system, residents of the municipality could save for a number of windmills, with a goal of generating thirty per cent of the town's electricity needs with their own windmills. This would require ten windmills. The idea originated with one of the members of the town council.

The townspeople were considerably less enthusiastic about the savings programme than had been expected. With great difficulty, they were able to save enough to purchase one-half of a windmill. To prevent the idea from becoming a complete failure, the municipality decided to supplement the amount, so that at least one windmill could be installed.

In the first draft of the final report, the engineer thus wrote that the result was quite disappointing. Such a report, however, would mean loss of face for the council member, who therefore urged a reformulation. The text ultimately came to read as follows: 'The project is a great success; the municipality has demonstrated its support for the environment and has made its – however modest – contribution to the fight against climate change'. The young engineer was initially unaware of the political framework of this project. In order to prevent future projects from the council member from getting off the ground, he was forced to play (along with) politics.

It is more difficult to carry out a project in a political environment than it is to carry one out in the environment of the sales representative. Decisions surrounding a project depend upon the political game and not on what would be most effective for the project. The catalyst for beginning a project is often political, and it therefore determines the force fields with which the project team is confronted.

Because of a re-organisation, it was necessary for a number of organisations to merge and cooperate. This re-organisation was mandated from above and involved, among other things, replacing a number of small-town local affiliates with a central office in the region. This meant that employees would have to travel much farther to their work. The work itself changed as well; many fewer positions for highly educated workers were available after the re-organisation. A portion of the personnel had to seek positions outside of the organisation or to other positions that were substantively much less interesting. There was therefore considerable resistance to the re-organisation, even though it would mean considerable

improvement in service for the customers if it proved successful. Finally, the re-organisation was to be carried by the personnel themselves, under the supervision of a project leader.

The project leader initially had difficulty getting the project started. The team members who were to carry out the work kept finding excuses not to do their jobs. There was always a problem or setback, and there was considerable discussion. The discussions usually shifted to the question of whether the project itself was a good idea. The project leader would then defend the project – it would mean a great improvement for the customers – but he was unable to generate any enthusiasm for the project.

Once the project leader realised that many of the workers did not (fully) support the project, he decided to focus first on the task of reducing resistance to the project. He accomplished this by taking the time to visit the various affiliates. He also talked more with the supervisors and employees, often informally at the coffee machine. By developing a better relationship with a number of the formal and informal power-holders, he was able to kick-start the project when it faltered. The project remained difficult, but the political approach worked much better than had the rational approach that he had tried initially.

A guide for playing politics would exceed the scope of this book. In short, the political game often takes place at the level of relationships and power relations. In a business environment, the product itself is more in the foreground.

Project leaders should realise that projects that are carried out with social organisations *always* involve at least some element of politics. To make projects successful, project leaders in this situation would be wise not to detach themselves from the political game. Instead, they should seek to play it as well as possible while providing substantive direction to their projects.

5. Waterfall versus cyclical project management

The six-phase model is a waterfall model. In other words, the phases take place in succession. Just as it is impossible to swim upstream against a waterfall, the pure waterfall method does not allow returning to a phase after it has been completed. During the implementation phase, it is not desirable to decide to adapt the design, thereby bringing implementation to a standstill. For a number of reasons (see e.g. McConnell, 1996; Kroll, 2004; Chromatic, 2003; Stapleton, 2002), the waterfall method is usually less suited to software-development projects.

- Software development is a creative process.
- It is nearly impossible to identify all of the requirements (functionalities) beforehand.
- Estimating the amount of time that will be necessary to implement a functionality is quite difficult.
- It should be possible for all intermediate results to be tested by users throughout the entire trajectory of the project.

Software development is a creative process

To outsiders, software development appears to have more to do with engineering than it does with inventing. It does, however, correspond to inventing in a number of ways. In the process of invention, one never knows in advance precisely what is going to happen.

The designers and programmers who write new software must conceive of solutions for the problems that are set before them. Regardless of the many methods and prescriptions for work, writing programming code remains largely new and therefore largely uncertain. Programmers can choose their solutions out of millions of possibilities that are written in dozens of programming languages, and which operate according to thousands of hardware configurations and in dozens of software platforms. Although this freedom does offer a number of advantages, it also makes the project more difficult to manage than traditional projects (e.g. construction or building projects).

It is nearly impossible to identify all of the requirements (functionalities) beforehand

The waterfall method prescribes the formulation of requirements as the project result of the definition phase. Ideally, there should be little further deviation from these requirements throughout the rest of the project. The development of software according to the waterfall method requires the investment of considerable time in the beginning of the project in analysing the necessary functionalities (requirements). This leads to a functional design (for more details on functional designs, see [i]). Using the functional design as a guide, the software architect makes a technical design in the design phase. The technical design includes a description of how the desired functionalities should be implemented.

Although this may appear quite straightforward, consider the following situation (example adapted from McConnell, 1996, p. 138). There is a project to produce a new automobile. As an active automobile driver, you are asked to formulate the requirements for an automobile. You consult with other drivers and create an extensive list:

- The automobile must accommodate four people.
- The automobile must have a steering wheel in the front on the driver's left-hand side, a brake pedal, a gas pedal and a hand brake.
- The automobile must have four wheels.
- The automobile must have white headlamps and red tail lamps.
- et cetera (the actual list would obviously be enormous)

Using your list as a guide, the designers develop a new design, which is subsequently built several months later. One day, as you are walking down the street, you see an automobile stopping. You realise that you neglected to include brake lamps in your list! You immediately telephone the engineer of the automobile manufacturer, who nearly explodes in reaction to your call. You maintain that it is only a small change. For the automobile manufacturers, however, it is a disaster. The automobiles that have already been made must be completely disassembled, cables must be stretched from the brake pedals to the rear, the rear of the automobile must be completely re-designed in order to accommodate the brake lamps, the boot hatches that had already been produced would have to be discarded and the list goes on.

While the example above is somewhat absurd, it reflects an almost daily reality in software development. Programmers erroneously assume that the clients, customers or users of the software that is to be developed know precisely what they want Clients erroneously assume that the software builders can make (and adapt) everything in the shortest possible time. This problem is the greatest source of conflict between customers and software builders.

The following anecdote illustrates the fact that there are many conflicts between customers and software suppliers. A beginning entrepreneur wanted to obtain legal insurance for his business. He asked about the possibilities. When the broker asked him to identify the sector in which his business was active, the entrepreneur replied, 'IT'. 'Too bad', replied the broker, 'there are so many lawsuits between IT suppliers and customers that there are no insurance companies that will write legal-insurance policies for IT companies'.

Estimating the amount of time that will be necessary to implement a functionality is quite difficult

The waterfall method assumes a number of phases. In their project plans, project leaders must include estimates of the amount of time (and therefore money) that will be needed for each phase. We have already seen that time estimates are generally difficult for any project, particularly if they must be made in the early stages of a project. For software projects, it is simply impossible. Imagine that it were feasible to compile a qualitatively adequate list of functionalities in the definition phase. In theory, the project team should then be able to provide a reasonable estimate of how much time will be necessary to implement each functionality. In practice, however, there are too many uncertainties to allow a reasonable estimate (e.g. McConnell, 1996):

- Once a functionality has been made, it is often discovered that the customer does not need it after all. The hours that were used in implementing this functionality can therefore be regarded as useless.
- Requirements change during the project.
- Should the functionality be implemented expensively or inexpensively? There are many possible methods of implementation and realisation.
- How will the functionality be designed technically? The design largely determines the amount of time that will be needed to make it.

- How high must the quality of the functionality be? For example, should a web application always remain completely available, or can it be offline for a few hours each year?
- The time needed to identify and repair errors in software can vary from minutes to weeks.
- How long will it take to install and integrate the new software into the customer's existing systems?
- The lack of knowledge concerning possible solutions also complicates the estimation of time. Further, it is difficult to estimate how long it will take to acquire the necessary knowledge.

The list above shows that many factors can affect the amount of time that will ultimately prove necessary to implement a new piece of software. Research (McConnell, 1996, p. 168) has shown that the estimates of the time needed to implement a functionality at the beginning of a project varies between four times too little time and four times too much time. This means that the amount of time necessary can turn out to be either four times higher or four times lower than a faulty estimate. These estimates become more accurate as the project progresses. After the functional design has been made, the margin is reduced to twenty-five per cent too much or too little. Only when the functionality is implemented can an estimate be provided with a high level of certainty (see Figure 8).

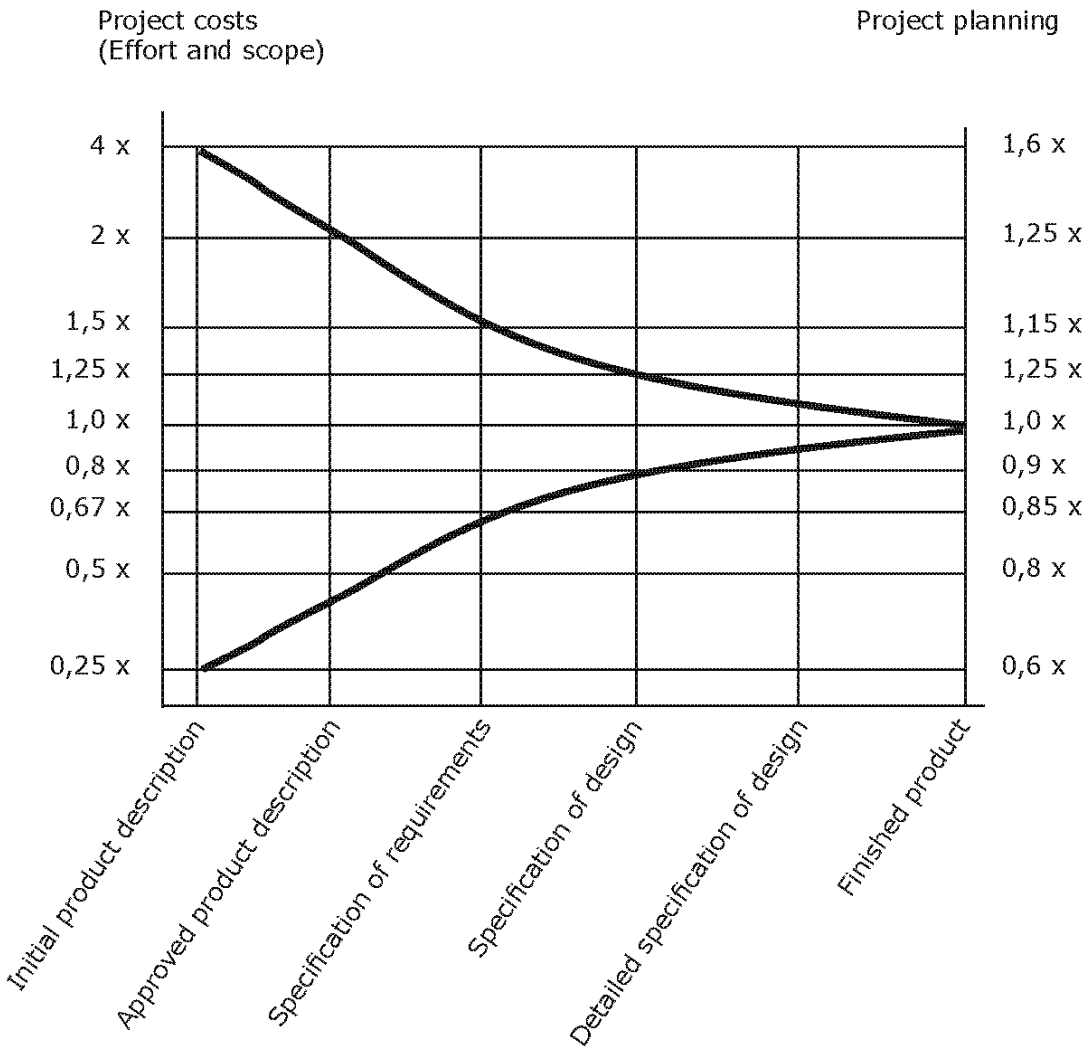


Figure 8: Accuracy of estimates of time necessary to implement a functionality (Source: McConnell, 1996, p. 168).

Software is never completely free of errors. Even for the well-known packages that are used by many (e.g. Word, Excel, Explorer, OSX, PHP, Flash), there are lists of known bugs available on the Internet. New releases regularly appear on the market, in which software errors have been removed. Customers sometimes expect error-free software. In practice, however, such a goal would make it impossible to complete a piece of software. Moreover, software errors are often not inherent in the software itself.

An educational game was made in Flash. It was agreed that the game would be delivered as a file and that the customer would install it himself on his own server. During the project, the customer requested that a high score feature be included in the game to increase competition between players. This would require a piece of script code in PHP. The game builders made and tested the script code on their own server, which worked in Linux. It worked. The game and script code were delivered to the customer. The customer, however, had a Windows server and, for some reason, the script no longer worked well. The high scores were not saved. The programmer eventually needed a week to resolve the problem. As it turned out, the combination of PHP and Windows does not always work well. The script itself

contained absolutely no errors! By using a hack, he was able to get it to work, and everyone was satisfied – but who should pay for that extra week of work?

Another software-development project also involved educational software. The problem was that the software worked for the programmers, but it did not work well for the customer. After trying nearly everything, the programmer decided to pay a visit to the customer. As it turned out, the customer was using an old Pentium III system. The slowness of the computer caused the poor performance of the software. The programmers had also tested the software on a Pentium III, but it had worked fine. Further investigation revealed that the customer's computer was so slow because it was full of viruses and spyware.

The uncertainty that is illustrated by the examples above does not simplify the writing of project plans. It also complicates agreements between the parties involved. Someone must assume the risks for extra work that must be done. If payment is on an hourly basis, the customer assumes the risk. If a fixed price has been agreed (as in grant-funded projects), the software builder assumes the risk. When more than two parties are involved, it becomes even more complicated. In such a case, who should pay for the extra hours in the project?

Discussions often arise concerning who should be responsible for delays. In many cases, the guilty party is difficult to identify. It is quite possible that none of the parties involved is at fault, as the 'delay' is actually the result of a faulty initial estimate of the number of hours that would be needed. Exceeding the number of project hours and the question of who should pay are frequent sources of conflict in the IT world.

It should be possible for all intermediate results to be tested by users throughout the entire trajectory of the project

In the definition phase and the design phase, customers are asked to formulate their requirements as well as possible. This is difficult for two reasons. First, customers have only a limited conception of the possibilities or impossibilities of IT. They do not have a good idea of what is or should be possible or what they should or should not want. Second, customers often have only limited knowledge of their own business processes. Many IT projects involve the computerisation of a customer's existing business practices. Even though customers may have worked with the processes for many years, they are often not able to define their own business processes. They can work fine in their own way, but cannot say exactly what that way is. The precise definition of processes is a precondition for making software that will drive computerisation. The complexity for customers thus increases if they must describe existing processes.

The list of requirements that is developed in the definition phase is often incomplete. In the implementation phase, programmers make software according to this incomplete list. When users are confronted with the initial versions of the new software, additional requirements are identified. 'It looks good, but now can you make it so that I don't have to keep entering my password?' Programmers often complain that customers do not know what they want. Customers appeal to the 'fact' that, because software developers are professionals, they should have determined earlier in the process what the customers wanted.

In a software project that involved the automatic processing of applications for a web-based service, an extensive functional design was made. Long lists of requirements from the customer were compiled. A number of screen designs and flow charts were added, after which the programmers could get started.
Probably because the project was under extreme time pressure or perhaps because the customer's organisation was rather chaotic, the designers had neglected to

include one important component: manual administration. The applications were processed by the software. Because the processing of the applications was to be automatic, the programmers thought that no manual administration would be desired. This requirement also did not appear in the functional design. When the software was delivered for testing, the customer realised that there were exceptions in some of the applications. These applications could not be processed automatically, but would have to be handled manually. The software, however, worked only automatically.

In the waterfall method, the actual project result is tested at the end of the implementation phase. This is late in the development process. The time between the definition phase, design phase and implementation phase sometimes amounts to months or even more than a year. If design errors are discovered in a late stage of the project, it can be expensive or sometimes even impossible to adapt software without beginning an entirely new project. Because we have seen that it is practically impossible to specify all requirements beforehand, a working method that allows the possibility of testing (intermediate) results earlier is desirable.

Comparing a number of prospective software houses, a customer asked the competing parties what was possible. One party was somewhat reserved and doubted whether some of the customer's requests would be feasible. The other party had an aggressive sales representative. When the customer asked the sales representative if a particular request would be possible, the sales representative telephoned his programmers. 'Can we build a functionality that can do X?' The programmer, who thought primarily in technical terms, answered that, in principal, anything was possible. Neither the programmer nor the sales representative worried about feasibility in terms of project management (e.g. time, money, complexity, risk).

The sales representative's enthusiasm was more effective than the other party's reserved attitude had been. The customer chose the aggressive sales representative's offer. The newly acquired project subsequently came into the hands of a project leader and a group of programmers. After a time, it became apparent that the project did not fulfil the customer's expectations. This had partially to do with the fact that the processes were much more complicated for the customer than they had originally appeared. During a heated discussion between the two parties, the customer referred to the fact that the sales representative 'had said that functionality X would not be a problem'.

Cyclical methods of project management

Because of the issues that have been sketched above, a number of other methods of project management have emerged in recent years. These methods are particularly suited for IT-development projects. Examples of these relatively new streams within project management include DSDM, RUP, eXtreme Programming (XP), RAD and agile project management (McConnell, 1996; Kroll, 2004; Chromatic, 2003; Stapleton, 2002, [ii], [iii])

Although the above-mentioned methods of project management differ according to a number of aspects, they are essentially the same. Because the path toward the final goal of IT projects has proved so uncertain, these methods assume that the goal will be achieved in a number of short cycles. This is the background for the term 'cyclical' project management for these methods.

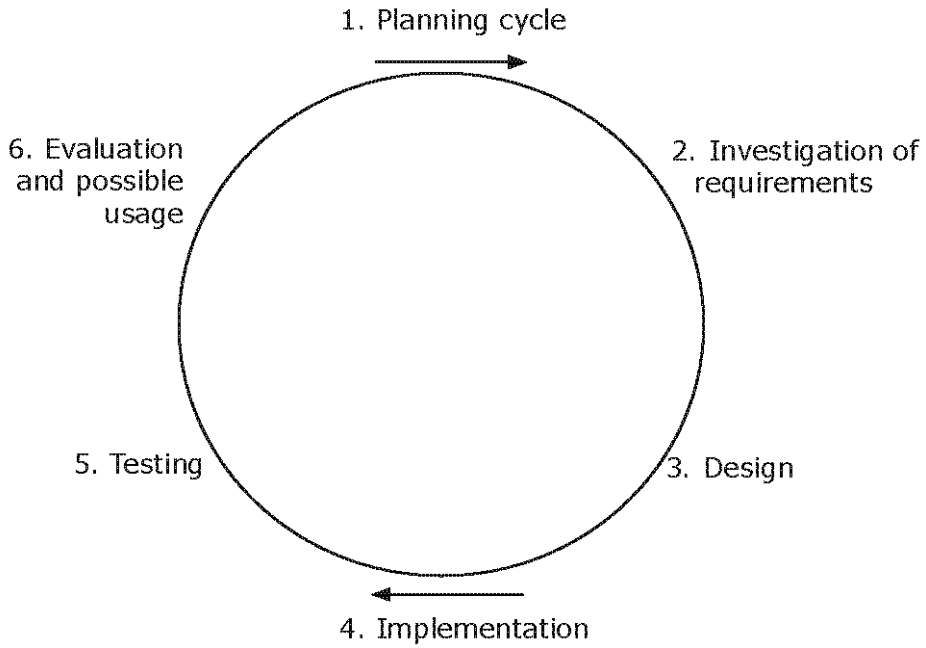


Figure 9: The activities in a cycle

In cyclical project management, the project goal is pursued in several short, successive consecutive cycles. Each cycle is relatively short, preferably lasting less than one month. Within each cycle, a portion of the project is carried out. Analysis, design, implementation and testing occur within each cycle. This is fundamentally different from the waterfall method, in which these activities all take place within their own separate phases. In addition, the waterfall method prescribes only single moments for definition, design, implementation and testing. In the cyclical method, this occurs many times in sequence.

Various components of the software are implemented during the cycles, which are therefore independent of each other. Evaluation takes place after each cycle is completed. Should advancing insight lead to new of different requirements for the project, the activities of the subsequent cycles are adapted to take them into account.

A cycle begins with the making or adjusting of the schedule. Next, the requirements of the result of the cycle are examined. A design is made, programmed and tested. Evaluation subsequently occurs and, in some methods, the new software is brought into use. Thereafter, the following cycle can begin, in order to carry out the following component of the project. (For a more extensive description of cyclical methods and the differences between the methods, see e.g. Kroll, 2004; Chromatic, 2003; Stapleton 2002.)

The most important advantages of working with the cyclical method are as follows:

- Higher product quality and improved implementation of functionalities,
- More realistic estimates of time and money,
- Project team is under less pressure,
- Higher quality.

The previous chapters have shown that it is difficult or impossible to determine of the desired functionalities accurately in the early stages of a project. In cyclical methods, the desired functionalities are implemented in several short cycles. In each cycle, a small portion of the desired functionality is not only investigated; it is designed, implemented and tested as well. The short succession of design, implementation and testing makes a particularly important contribution to quality improvement. Teams are thereby in state to make adjustments. If a design does not turn out to be good in practice, it becomes obvious during the cycle, thereby allowing adjustment. This way of working also allows customers to request adjustments.

Another reason that cyclical project management leads to better quality is that a cycle involves intensive collaboration between customer, designers and programmers. A multi-disciplinary team jointly conceives of and implements solutions. In the waterfall method, the customer is involved primarily in the beginning, in the formulation of the requirements; thereafter, the designers make a design and then the programmers programme the software. The project leader serves as the link between all of the various parties and must ensure that information that is exchanged is delivered to the right place. In practice, many programmers and designers never see the customer, who re-emerges in the process only after the software has been completed.

Cyclical methods of project management are particularly suited to projects in which the goal that is to be achieved cannot be clearly established beforehand, as in creative projects or research projects. Working in a number of cycles with a multi-disciplinary team in which the end-users are also represented allows the team to discover the real goal of the project and how it can best be achieved. Each cycle contains a point for reflection and an opportunity for adjustment.

In waterfall projects, a goal is formulated beforehand. Reflection on the original goal is less of a possibility. The originally formulated goal is never (fully) achieved, and it is likely that neither the originally formulated goal nor the goal that is achieved is exactly what was originally desired.

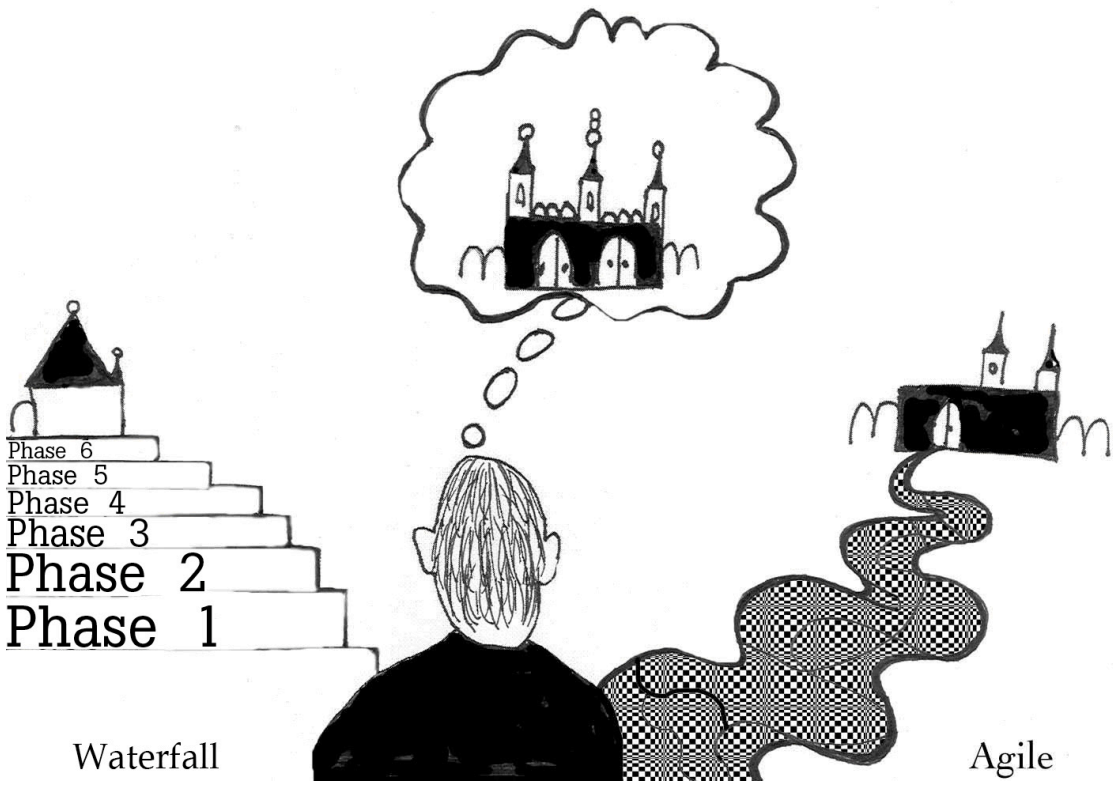


Figure 10: Better results are achieved by working in cycles. (Illustration: Rachèl Harmsen)

The last reason why cyclical project techniques generate better results is that the customer performs acceptance tests. Quality is further strengthened by using tests as criteria for well-performing functionality from the very beginning. Programmers must therefore define 'failing tests' (or unit tests) before they write their code. The code is considered acceptable only if it passes the failing test.

Test-oriented work requires programmers to prove that there are no bugs in the new code before they write new code. They do this by developing a test (failing test) that would detect any possible bugs before they begin coding. For example, imagine that a programme must be written for calculating the correct amount of change that you must receive from a candy machine. First, the availability of a function to give change must be tested. This function could be called 'give_change'. A simple test could be made and, when it is performed, it is determined that a 'give_change' function does not yet exist. If the programmer then makes the function but does not yet give it any substance, the test will succeed.

The next step would be to test whether the machine gives the right amount of money back when an item is purchased. Insert sixty cents into the machine and buy an item that costs fifty cents. The test will not succeed, because the function is still empty. The programmer then writes the code. In the 'give_change' function, he writes that the amount of change to be returned is the amount of money that was inserted in the machine, less the price of the chosen candy. The test should now succeed. The process continues in this way; for each component of the software, a test must be devised beforehand (example translated and adapted from Chromatic, 2003, p. 27 ff).

The code is not the only component that must be technically tested; the functionalities must also be tested (i.e. acceptance tests). For these tests, the customer determines the conditions under which the functionalities that are to be built can be accepted, before coding begins (e.g. how quickly a computer must respond to a certain action of a user). The prior specification of the conditions under which the new functionality can be accepted has proven particularly difficult and time-consuming. Nonetheless, the active role of customers in testing is an important determinant in the success of a project.

More realistic estimation of time and money
With cyclical project techniques, the functionalities that are to be implemented are not written in stone at the beginning of the project. The available time is specified. Agreements are made concerning the direction in which the project will proceed, and it is determined in the process what is actually needed, useful and realistic with regard to the software that is to be made. In cyclical projects, the functionalities that are to be implemented are not established as fixed goals, and they thus avoid the risk that the necessary hours, and therefore money, can get entirely out of hand. To prevent such a situation, the available time is used as a starting point, and it is determined during the process what is realistic to expect in that amount of time. Also for this reason, cyclical methods of project management are much friendlier for the project team. The team does what it can do within the specified time, but is not pressured to meet unrealistic schedules or to work within an inadequate budget.

Cyclical methods also facilitate the management of external suppliers. With the waterfall method, a project can become bound to a single supplier until the end of the project, regardless of the performance of that supplier. In the cyclical working method, is theoretically possible to make new agreements for each cycle or even for each component to be delivered and, if necessary, to change suppliers.

Conditions for cyclical project management

To apply cyclical project management, a number of conditions must be met:

1. Users/customers are actively involved.
In cyclical project management, the formulation of requirements, design, implementation and testing take place in each cycle. This means that many decisions must be taken in a cycle. If the software is to provide a good reflection of the wishes of the customer, the customer *must* participate actively in the project team. Customers must present their wishes as clearly as possible to the programmers and designers. This generally involves weekly (or at least bi-weekly) presence in the project team.

Within a project, customers are involved with determining the desired functionalities and the planning of the cycles. They collaborate on the acceptance tests, approve or reject intermediate results and collaborate on the general direction of the project. The active involvement of the customer also leads to better results in the waterfall method.

2. The team is authorised to take decisions.
Within a cycle, the project team must be authorised to do what they think is best. If the project team does not have this power, the cyclical model of project management will not work. If constant authorisation from superiors is necessary during a cycle, this can lead to stagnation. Moreover, outsiders are often poorly informed about what is going on, because they do not actively participate in the project team; this makes it difficult for them to make sensible decisions.

3. Project result (software) can be broken down into smaller parts.

With cyclical project management, parts of the project are performed in a number of cycles. This is possible only if the software that is to be created is divisible into a number of more or less separate components.

4. The requirements that are imposed by management with regard to the software are primarily global; management does not impose direct, concrete or specific requirements. One of the strengths of cyclical project management is that the customer, designers, programmers and any testers work closely together within the cycles. If there are already specific and concrete requirements at the beginning of a project, this impedes the freedom of the project team to use their better judgment to make design choices. Many requirements on a project are revealed during the process to be in need of adaptation and should therefore not be (too) firmly established in the beginning.

5. The activities are intelligible for the customer.
If considerable technical work that is difficult for the customer to understand must take place within a cycle, a risk emerges that the customer will not be in state to participate well in the team. In such a situation, the customer has very little to contribute to the necessary design choices.

A similar risk arises when the progress is not visible to the customer. For example, much of the work may involve coding, with very little being done on the user interface. It is important that customers have sufficient insight into the substance and progress of a cycle in order to ensure that they are not pushed into the background.

6. It should be possible to take a step backwards.
Even in cyclical project management, teams sometimes pursue paths that later prove to have been wrong. In such a case, it should be possible to take a step backwards. If a new module that is created in a cycle proves inadequate, it must be possible to resume working with the old module. This imposes demands, particularly with regard to the archival and documentation of the project materials. Concurrent Versioning System (CVS) and Subversion are two helpful tools for these tasks (see Appendix 3 for a list of tools).

7. In addition to programming, programmers should be able to communicate with customers, and vice versa. Team members must be in state to think conceptually. Discipline is necessary in order to persevere with the work.

8. The organisation in which the project takes place must also offer sufficient support for this method of working. Systems for time reporting, archiving and scheduling are necessary to support the projects. These registration systems ensure the transparency that is necessary to ensure the adequate distribution of resources across projects and time.

9. Projects should have sufficient priority, team members must be released for projects. Requiring team members to work in too many projects at the same time does not work. If an organisation is insufficiently adjusted to working with projects, the flexibility of cyclical project management is likely to lead to chaos. The waterfall method also benefits from an organisation that is arranged for working with projects (see Wijnen, 2004, p. 111).

The director of a software house, who was more of a visionary than he was a manager, had a brilliant idea nearly every month, and he was continually starting new projects in his company. Older projects were consequently never completely

finished, and the employees were sometimes working on as many as five projects at once. The charismatic director had once read a book on rapid application development (RAD) and was quite enthusiastic about it – particularly about the aspect of 'rapid'. He posted the basic concepts of RAD over the copier and subsequently assumed that everyone would start working according to these concepts. After all, it was a wonderful method.

Risks of cyclical project management

Cyclical methods of project management sometimes allow insufficient time to implement the desired functionalities. Because the amount of time is pre-determined, fewer functionalities will probably be made than were originally assumed.

This is indeed a real risk, but it is also inherent in the waterfall method. In the waterfall method, the definition phase includes an extensive analysis of requirements. This analysis could be expected to generate better time planning. This is often not the case in practice, however, for the reasons that are mentioned above. Functionalities are left out in this method as well, as there is not enough money to implement them.

In cyclical methods, requirements are handled pragmatically. For example, the requirements in cycles can be divided according to the MoSCoW rules (Stapleton, 2003):

- Must Have: requirements that are essential for the system
- Should Have: important requirements that people really want
- Could Have: requirements that are desirable, but which can be easily omitted
- Want to Have: but will not have this time round: requirements that can wait until later

Regardless of the fact that there may be no more time for particular functionalities, the DANS project managers agree that cyclical work yields more customer satisfaction than the waterfall method does. At any rate, the expectations are consistently better managed, and the projects deliver results that actually work, even if they are less elaborate than originally hoped.

One frequently mentioned disadvantage of XP is that considerable power comes to rest with the programmers. If they misuse this power, they can hide behind the customer's lack of technical knowledge. Preventing this situation requires a project leader who can serve the interests of both the programmers and the customer. Such project leaders assist their customers in choosing and planning the cycles, making the technical background of choices intelligible and providing for administration and reporting.

Finally, another disadvantage of XP is that there is a steep learning curve for programmers, managers and customers with regard to the introduction of the method.

6. DANS software-development method

The penultimate chapter of this handbook for project management provides a sketch of the method that DANS applies for the management of software projects.

One frequently mentioned disadvantage of cyclical working methods is that they require teams to start working immediately. Too little consideration is given to what exactly is desired. The expectations of possible customers or clients are not managed well. Agreements concerning the desired results are inadequate. In this respect, cyclical methods are less advantageous than is the waterfall approach, in which all of these matters are settled in the beginning.

In an effort to avoid this dilemma, DANS applies the best of both methods for its software-development work. Projects begin with the waterfall method, so that adequate consideration is given to requirements, requests and design. After the design phase, there is a shift to the cyclical method, thus allowing felicity for handling these elements. The cyclical component of the DANS method makes use of eXtreme Programming (XP) (Chromatic, 2003, [ii], [iii]). Further definition, design, implementation and testing takes place within the cycles. Once the software is sufficiently developed, the follow-up phase begins. Each step in this working method is described below.

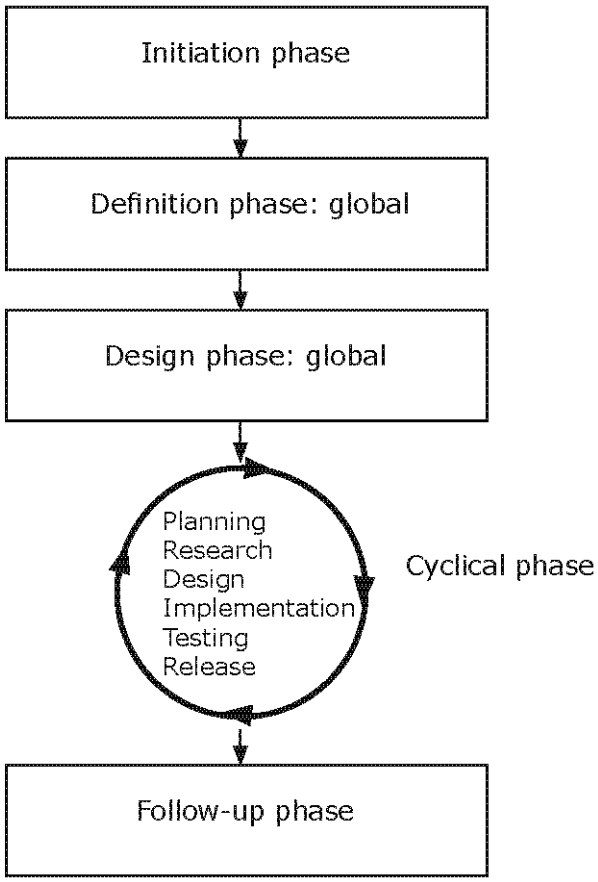


Figure 11: Schematic illustration of the DANS software-development method

Initiation phase

The initiation phase begins with an idea for a project. No budget is yet available for the project. The goal of this phase is to write a project plan according to which internal or external financing can be requested.

Activities in the initiation phase:
- Elaborate the concept.
- Examine the base of support.
- Contact possible partners.
- Investigate funding opportunities.
- Prepare an initial global estimate of the control factors for the project.
- Prepare a concrete estimate of the control factors for the definition phase.
- Establish project boundaries.
- Prepare a project plan.
- Apply for funding or establishing contract agreements with possible customers.

End result of the initiation phase:
- Approved and funded project plan
- (Possible) contract with customer

Operations/Decisions:
- (Prospective) project leader
- Client
- (Possible) customer

Tools:
- See Appendix 10 for a model of a project plan.

If possible, instalment financing is preferable to lump sum financing following the initiation phase. For instalment financing, a relatively small amount of funding for the operations of the definition and design phases is requested during the initiation phase. Depending upon the outcome of these phases, a second request for funding for the rest of the project is made at the end of the design phase.

Definition phase

After a project plan has been approved, the project enters the second phase: definition phase. In this phase, the requirements that are associated with the result of the project are determined as clearly and as completely as possible. This is in order to identify the expectations that all involved parties have for the project result. The list from Chapter 1 can serve as a memory aid in this regard:

- Preconditions
- Functional requirements
- Operational requirements
- Design limitations

The collaboration of all parties that are involved in a project is very important in the definition phase, *particularly* the end users who will actually use the project result.

Activities in the definition phase:
- Compile list of requirements together with client, (possible) customer, end

users, experts and project team.
- Balance requirements.
- Test the feasibility of the requirements.
- Report to client, customer or both about the requirements.
- Report on the control factors that have actually been implemented thus far.
- Prepare new global estimate of the control factors for the rest of the project.
- Prepare a concrete estimate of the control factors for the design phase.

Result of the definition phase:
- Approved (tentative) list of requirements
- Approved control reports and prognoses

Operations:
- Current or prospective project leader
- Client
- Current or potential customer
- End users
- Experts
- Current or future programmers and designers
- System administrator (in connection with requirements in the follow-up phase)

Decisions/approval:
- Project leader
- Client
- Current or potential customer

Tools:
- See Appendix 10 for a sample project plan.

In theory, the waterfall method specifies that no additional or supplementary requirements can be added to the project after the definition phase. The list of requirements serves as the foundation for the contract between the project team and the client or customer. The list of requirements is the one to which the project result must ultimately conform.

Because this can cause problems with software projects, as we have seen, DANS does not apply this method strictly. We use the definition phase to investigate the requirements in order to provide the best possible direction for the project. Descriptions are also made of what *will not* be made, in order to clarify the expectations between customers and producers. Should the advancing insight that is acquired during the cyclical phases show that certain requirements must be re-defined, this method of working allows for adjustment (obviously in consultation and well documented).

Design phase

With the list of requirements that was developed in the definition phase, the project team is able to make choices concerning the ultimate appearance of the software. A design document is the result of the design phase in IT projects. The design document contains an elaboration of the concept and a broad outline of a technical design. The goal is to investigate what the software will look like, both technically and functionally (a sample functional design can be found in [iv]).
In this regard, it is helpful to work with dummies in the design phase. A dummy is a quickly assembled, non-operational (or only partially operational) piece of software that serves primarily to evaluate the design. These dummies are

presented to the clients, customers (or both), as well as the end users. One advantage of dummies over schemas on paper is that they resemble the finished product.

In theory, the waterfall method specifies that it is not possible to reverse any design decisions after the design document has been approved. This is possible in the DANS working method. The design document serves as the starting point for the builders. Should advancing insight show that changes in this document are needed, however, they can be carried out. In addition to being programmed, any design adjustments that are made during the cyclical phases must be documented in a 'project log'. Once the design is sufficiently elaborated (in the opinion of the project team), the cyclical phase can begin.

Activities in the design phase:
- Prepare the design document.
- Create and evaluate mock-ups (i.e. dummies) with the customer.
- Report on the selected design.
- Report on the control factors that have actually been implemented thus far.
- Prepare a new global estimate of the control factors for the rest of the project.
- Prepare a concrete estimate of the control factors for the cyclical phase.

Result of the design phase:
- Approved design document
- Approved control-factor report and estimates

Operations:
- Project leader
- Designers
- Programmers
- Current or potential end users for design evaluation

Decisions/Approval:
- Project leader
- Client
- Current or potential customer

Tools:
- See [i] for guidelines for creating a design document.
- See Appendix 10 for a model of a project plan.

Cyclical phase

The working methods in the cyclical phase are borrowed from XP. In this phase, a number of cycles are performed in succession. A cycle lasts no more than one to two weeks. The following activities take place within *each* cycle:
- Planning
- Examination of functionalities
- Design of functionalities
- Implementation of functionalities
- Testing of functionalities
- Delivery of functionalities

Planning
At the end of the design phase, an estimate is made of the number of cycles that will be needed to achieve the project goal. This occurs according to the functional

and technical design. Cycles are never long; they usually last between two and four weeks. A cycle always includes the activities of planning, investigating, designing, implementation, testing and delivery. Each cycle, therefore involves only a few functionalities (or sometimes only one).

The procedures for planning are as follows. The desired functionalities are written jointly by the project leader and the customer on 'story cards', starting with the functionalities that were determined in the definition and design phases. Using the story cards as a guide, the programmers create a task list, which is a list of tasks that are involved in implementing a functionality. These tasks should generally not last longer than a few hours. If a task takes longer, it should be divided into a number of sub-tasks, or the story card must be divided into several story cards. Story cards that contain too much work to be completed in one cycle should be returned to the customer, who must divide the requests and distribute them over additional story cards. The programmer estimates the amount of time that will be necessary for each task, thus producing a time estimate for each story card. Customers can now work with the project leader to determine which of the functionalities they would like to be implemented first in the next cycle.

How long does it take to create a website and fill it with fifty pages of text and a number of photographs? A quick answer from the designer is that it would take 'about two weeks' is much too rough. It could well take much longer. Dividing the task reveals the necessity of creating a CSS, consulting with the client about the design, programming the design in XHTML, shortening the texts for the Internet, filling the pages with the texts, adapting the photographs for the Internet and placing them, allowing the customer to examine the website and removing the last remaining errors.

Dividing the work into smaller parts reveals that the task involves much more work than was initially thought. The client also realises that he is also expected to do a number of things. Estimating the amount of time that is necessary for each task yields a much better total estimate (and shows that it is not possible within two weeks).

Once the programmers begin, they keep a record of the hours that they have used to perform each task. They often need more hours than they had originally estimated. Because they have the opportunity to refer back to their own estimates, the programmers learn to make increasingly accurate estimates.
Experience has shown that, as a project progresses, a relatively constant difference factor emerges between the estimates and the number of hours that are actually needed. After a few cycles, this factor, which is known as the 'drag factor', can be determined according to the average difference between the estimated and the actual number of hours. The drag factor is used for planning future cycles and in reports. Multiplying the number of remaining story cards with task lists by the drag factor provides a particularly reliable indication of how much time is needed to implement the rest of the project.

Because the number of hours for the project is limited, choices must be made. The availability of a reasonable estimate of the time needed to implement a story card allows a good deliberation between the various story cards. Some story cards are simpler to implement than others are, and some story cards may be eliminated. An important starting point for determining the order is that risky story cards must be handled first, in order to eliminate the most important risks as soon as possible. The MoSCoW rules are also applicable, or a simple prioritisation from one to five.

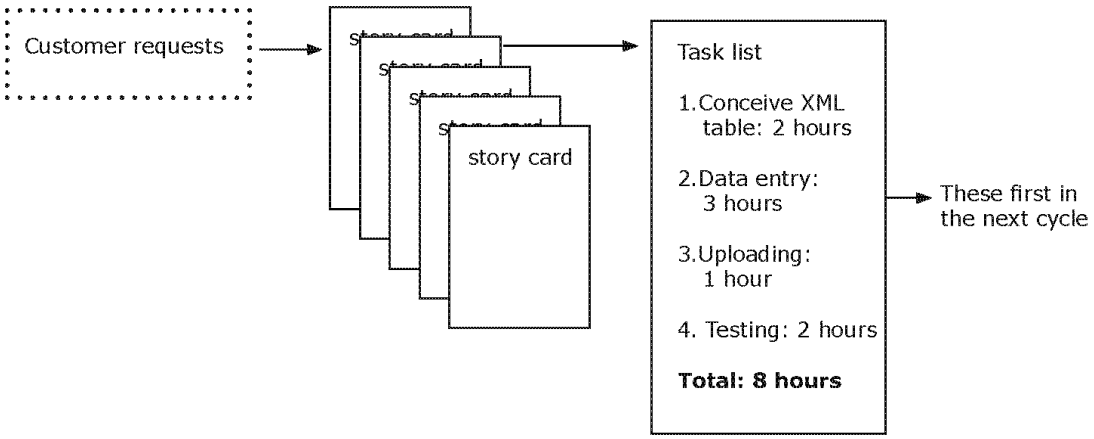


Figure 12: Planning functionalities in the cycles

Investigating and designing functionalities
The initial investigation of a functionality takes place during the creation of the task list for planning. By deciding ahead of time which sub-tasks will be needed to implement a functionality, the programmer gains more insight into the functionality itself. The involvement (presence) of the customer is essential for the further investigation and design of the desired functionality. Customers should specify exactly what they want. In the beginning, customers have much more contact with the programmers than they do later in the project, although caution must be taken to ensure that the programmers do not start thinking for the customer, thereby making erroneous assumptions.

Implementation of the functionalities
After the customer and the producers have agreed on a design for the desired functionality, it is built. Programmers always perform this work in pairs (in XP terms, this is known as 'pair programming'). Although this may seem non-productive, pair programming offers the following advantages:

- The knowledge of two programmers is combined.
- Less time is spent transferring knowledge or code within a project.
- Fewer errors occur in programming.
- Problems are resolved more quickly.

There is an additional advantage to working in pairs: the greatest problem is getting started with programming. Once the work has been started, it can proceed. Programming in pairs allows the programmers to encourage each other to get the work started.

Joel Spolsky, a programmer for Microsoft, realised that he worked effectively for only about three hours each day [v], and often even less. Other colleagues apparently had the same experience. The rest of the time was spent drinking coffee, reading e-mail, surfing the Internet, chatting with colleagues and looking at the beautiful office courtyard. Working with a partner can increase motivation, thus making it easier to get started.

Despite its advantages, pair programming also places considerable demands on the concentration of the programmers, and not all programmers like that. Further, not all combinations of programmers are capable of working together well. To minimise these disadvantages, a team might decide to use pair programming for more than half of each working day (for a further discussion of pair programming, see [vi]).

Testing and delivery
It is essential that every cycle culminates in the release of a new component of the software and that each component that is delivered is tested Testing consists of a unit test (conducted by the programmers) and an acceptance test (conducted by the customer). The customer is thus needed for this task as well.

The assumption behind including testing in each cycle is that it becomes exponentially more expensive to repair errors in relation to the time that it has taken to discover them. A basic assumption underlying the delivery of software in each cycle is that customers are able to see value for their money as quickly as possible and that programmers can receive feedback from the users as quickly as possible. Customers are able to see the progress of the project clearly. This is particularly important psychologically, and it can improve the customer's attitude considerably.

The working methods of DANS differ substantially from XP on one point: XP prescribes that a design may not be made before programming has begun. This is to achieve flexibility and avoid setting many things in stone that later prove less useful. The author and advisors who have prepared this handbook are of the opinion that it is indeed helpful to start creating a design in the definition and design phases. In contrast to the waterfall method, the DANS method allows the functionalities that were determined in these phases to be adapted in the cyclical phase.

Activities in the cyclical phase:
- Work through a number of cycles, each of which involves investigation, design, implementation, testing and delivery.
- Prepare story cards.
- Choose among the story cards.
- Plan the cycles.
- Ensure progress (control factors).
- Prepare a concrete estimate of the control factors for the follow-up phase (at the end).

Operations/decisions
- Project leader
- Client or customer
- Current or potential end users for testing and design
- Programmers and designers

Tools:
- See Appendix 3 for tools for recording and managing story cards and task lists.

Follow-up phase

After an adequate result has been achieved in the cyclical phase, the project enters the follow-up phase. In this phase, the project result is secured. What this means depends upon the type of project and on the agreements that have been made with the client or customer. For a research project, a final report would probably suffice; the development of a new product would require more follow-up.

Most of the problems in the follow-up phase arise because no clear agreements were made between the customer or client and the project team at the beginning of the project. The following are among the points that should be taken into consideration:

- How long should the follow-up last?
- What does the follow-up entail?
- How quickly must errors be repaired?
- Is there a guarantee on the project result?
- Who is responsible for bugs that are found after the project?
- Should documentation be delivered along with the project result?
- Will the users require training, schooling or both?
- Who is responsible for updates?
- Who will own the code, and who will be authorised to change it?
- Who will pay for the above-mentioned points?

It is important to realise that a project organisation is focused on temporary activities and is therefore not focused on offering (lengthy) support for the software that they have developed. Other means of support must be found for the longer term. Special (commercial) organisations exist for managing software, offering help-desk support, trainings, server administration, application administration and similar services. These organisations are likely to be (too) expensive for small non-profit initiatives.

Another alternative for securing the continuity of the software is to make it open-source. For this solution, an organisation is established to allow a group of developers and users to maintain and support the software.

Activities in the follow-up phase:
- Report on the control factors of the project
- Compile and submit final statement.
- Dissolve team.
- Transfer to the administrative organisation.

Result of the follow-up phase:
- Project statement
- Transfer documents

Operations:
- Project leader
- Team members
- System administrator

Decisions/Approval:
- Project leader
- Client
- Current or potential customer

Tools:

- See Appendix 10 for a sample project plan.

7. Programme management

Up to this point, this book has primarily addressed the management of single projects. This chapter delves deeper into the questions that arise when an organisation conducts multiple projects at the same time. These questions arise primarily in the area of relationships between the (higher) management team and the project leaders, and they are otherwise independent of the choice for a waterfall or a cyclical management approach.

Coordination of projects

As described in earlier chapters, the control factors are the parameters along which projects are reported on and directed. These factors also play an important role in the coordination of multiple projects:

Money: determining whether projects are financially feasible

Organisation: arriving at mutual agreements concerning the hierarchy among projects and between the projects and other departments

Quality: determining whether the goals of a project are consistent with the strategy of the organisation

Information: establishing who will report what about the project and when to the management team?

Time: estimating how many personnel will be needed within a given period to arrive at a good distribution of workers across the project teams.

Before the start of a project and after each project phase, a project leader should provide an estimate of the control factors for the rest of the project. The project leader also evaluates these factors as they have been implemented thus far after each phase. This information is transferred to a programme manager or the management team for decision-making purposes, usually in collaboration with the project leader and external parties (e.g. customers, financers). Several of the most important decision criteria are described below, particularly those that relate to the coordination of projects.

Money
The evaluation of financial matters by a programme manager involves the following issues:

- Is the project as a whole, and the following phase in particular, adequately financed?
- What are the possible financial risks of the project? Should a go/no-go moment be arranged?
- What is the liquidity prognosis for the project? Would a problem arise if the income from a project were to arrive later than the expenditures (e.g. if the subsidy is paid only after the completion of a lengthy project)?

By adding all of the budgets together, a programme manager can compile a yearly schedule for the projects. In this way, the programme manager ensures that there will be sufficient turnover from the projects to fund the project workers. Insufficient

turnover necessitates the solicitation of new projects. If there is too much turnover, extra personnel should be hired and/or projects must be postponed or even cancelled.

This process provides an idea of whether the projects as a whole are sufficiently profitable financially. For example, consider an organisation that is conducting ten projects of two thousand hours each. The organisation has twenty FTE of project employees in service. We will see later that almost all of the hours of the workers are divided. Assume that the annual salary costs for these twenty FTE amount to one million euros. Should the projects generate less than one million euros (in subsidies, internal financing, commercial proceeds), problem could arise. Fee adjustments or the cancellation of unprofitable projects may be necessary.
In projects, the actual amounts that are needed can vary greatly from those that are estimated in the budget, particularly if the project is subsidised (see Project reporting). To compensate for this type of uncertainty, a programme manager should reserve an amount to accommodate unexpected disappointments in one or more projects.

Budgets are also sometimes too broad. In such cases, it is important for project leaders to be prepared to give up some part of their budgets. Otherwise, any projects that did not remain strictly within their own budgets would be too expensive. If projects that are estimated too high assign part of their budgets to projects that were initially estimated too low, the organisation should finish (reasonably) even.

What should an organisation do if it has received a subsidy of €100,000 for a project that turns out to need only €80,000? The money will surely be spent; there are many other uses for that amount of money. For example, another project received €150,000 but actually needed €200,000. Fortunately, it is not possible to determine exactly what the programmers did in the hours that they charged to the projects. With a little re-arranging, everything will line up neatly again.

It is logical for organisations act in this way with their subsidised projects; they could hardly do otherwise. This is a direct result of the fact that organisations must compile a total budget for each application, from which they will no longer be allowed to deviate. One consequence of this situation is that the (financial) project statements from subsidised projects must be taken with a grain of salt. From the perspective of project management, it is unfortunate, as it makes it difficult to determine the exact costs of particular projects. It also contributes nothing in the area of accountability for the use of subsidy money.

Time

Adding all of the time estimates for all of the projects that an organisation will conduct in the coming period provides an indication of the workload for that period. For a given year, the number of FTEs should exceed the total number of hours from the project plans.

Consider the following example: Ten projects together ‘use’ 20,000 hours. There are twenty FTEs within the organisation, each involving 1700 hours of work. Seventy per cent of these hours are available for projects, and thirty per cent are assigned for general work (e.g. meetings, e-mail, travel time, other tasks, educational leave). This leaves a net of 23,800 (20 x 1700 x .70) hours each year that are available for projects. In this example, therefore, there is sufficient capacity (in global terms) to carry out the projects in the coming year. It would not be possible to add many more without hiring additional employees (and this would require a margin as well).

The calculations above are based on a global, annual perspective. Additional detailed information is needed concerning the necessary capacity, specified according to the tasks or roles of the project workers. The twenty FTEs must be further subdivided into various roles within a project (e.g. programmers, project leaders, designers, system administrators). Although there are probably enough employees, in general, to carry out the projects for the coming year, there may be too many project leaders and too few programmers.

Finally, the workload (expected number of hours) of the projects must correspond to the number of available employee hours for shorter periods as well as for the entire year. If all of the projects in the example that is described above were planned for the first three months of the year, that would cause a problem, even though there would be enough workers for the entire year. The operational distribution of workers on a monthly or weekly basis is accomplished according to the scheduling of the phases of the projects, or time boxes for cyclical projects. In addition, weekly or monthly consultation between the programme manager and all of the project leaders is necessary in order to create operational schedules for the employees.

The following are frequent complaints within project organisations with regard to time:

- 'There is no clear picture of the work pressure in this organisation. Sales and management keep adding new projects even though we are already having trouble carrying out the projects that we already have.'
- 'If one project goes off schedule, it has many consequences for other projects. Because we must share resources, a delay in the first project causes delays in all the other projects'.

The first complaint is common in organisations that are growing quickly and that are oriented toward generating as much turnover as possible By implementing the control mechanisms that are described above, an organisation can gain insight into its capacity on an annual basis, as well as its operational capacity from month to month or even from week to week.

The second complaint has a number of causes. First, in organisations in which this complaint is common, people are often not capable of closing out old projects. Each time that a project is to be completed, the customer makes a new demand, a new bug is detected or the project is expanded. To prevent this situation, it is important that agreements concerning when the project will be completed are made as clearly as possible in the *beginning* of the project. Further information about addressing this problem is provided in the other chapters of this handbook. (Excessive) work pressure is also a possible cause of the second complaint. Because the organisation is so busy, it may allow no margins between projects. This can cause a slight delay in one project to have a direct impact on subsequent projects. Finally, the second problem is probably less common in organisations that use cyclical project management, as they work with fixed time boxes.

In an effort to increase productivity, personnel are often assigned to too many projects. An attempt to fill the lost hours with other projects, however, can cause serious delays, as illustrated in the following example (borrowed and adapted from Goldratt, 2002):

A programme manager must carry out Project A, which consists of three tasks (a), in addition to Project B, which consists of three tasks (b). Each task requires five time units. If the projects are carried out in the order 'aaa, bbb', project A will be completed in fifteen (5+5+5) time units. Project B will also be completed in fifteen time units, measured from the moment at which Project B begins.

Another image emerges, however, if the tasks are not performed in direct succession (i.e. they are performed in an alternating sequence). If a task (a) is performed in alternating succession with a task (b), the order of work is 'ababab'. Projects A and B will now require twenty-five time units (5+5+5+5+5) each for completion. The fact that switching between projects also takes time has not been considered.

According to Goldratt (2002), the fact that organisations often assign personnel to many projects simultaneously is one of the most important reasons that projects take (much) too long. Projects that are slated for completion within three months often actually last more than two years. If the projects had been accomplished one after the other, each would have been accomplished in about three months.

This example, along with waiting-list theory, shows that it is not sensible to place such a heavy load on the personnel. Because of short-term cost considerations, management teams are primarily focused on having people work as much as possible. This causes projects to lose considerable speed. It is safe to assume that increasing the utilisation factor by ten per cent can increase the average turn-around time of projects by forty per cent. The costs of these delays, however, are much less visible, particularly in non-profit organisations.

Time registration

At the beginning of a new project, the programme manager must perform the controls that are described above with regard to finances and time capacity. This must be done again between the various phases of a project, with appropriate budget adjustments. Foresight is not the only reason that this is important, however; it is also necessary to evaluate the budgets (of time and money) that are actually used. Were the projects indeed profitable? Were the costs covered as originally estimated? Did that one project actually need 450 hours, or did it turn out otherwise? These questions must be asked and answered in order to improve the quality of project work in the future.

The ability to perform this evaluation requires the availability of a time-registration system. This system is also desirable for projects that are managed by project leaders. Within organisations, there is often resistance to time-registration systems. People feel as if they are being controlled, and they despise administrative chores. Another source of resistance is probably the fact that time-registration systems make it immediately obvious that (often) very little progress has been achieved. The implementation of a time-registration system is a project in itself.

One important argument for the introduction of a time-registration system is the transparency that it can provide. Employees (often) complain of excessive work pressure. The use of a time-registration system that includes both time registration and time reporting makes the work pressure visible. As soon as management or the sales department cause a project to 'go over budget', the project team can show in black and white that it is already fully booked for the coming period, and that the new project will have to wait.

What requirements must a time-registration system meet?

- A time-registration system must be used throughout the entire organisation, even by the bookkeeping department.
- A time-registration system must be accessible everywhere (e.g. web-based).
- Project leaders must be able to see time reports quickly (i.e. there should not be more than one week between time registration and the internal report).
- (Time) schedules must be integrated into the time-registration system.

- Time-registration systems must allow hours to be charged to work packages in the various phases of projects It is not sufficient to charge hours to general project names.

Project leaders' heaven
For many project leaders, it is difficult to keep the team members within the budget. Operational personnel apparently pay little attention to the clock; even if it has been agreed that a particular chore should not last longer than a few hours, there is always a reason why it nonetheless took more time.

One architectural firm addressed this problem by reversing the responsibilities. The firm's technical artists were required to act as internal freelancers, soliciting work from the various project leaders. This created internal competition. An artist who claims to be able to complete a job in less time than another artist has estimated will probably get the job. Furthermore, eight hours is eight hours. Artists must therefore use their own time to complete jobs that are not finished in the agreed-upon period. It would be wonderful to be a project leader in this organisation; it would probably be less attractive to project employees.

Organisation of multiple parallel projects
Wijnen (2004, p. 187 ff) distinguishes among three structures of project management with regard to the mother company:

- consultation or coordinating structure;
- matrix structure;
- pure project structure.

In the consultation or coordination structure, the 'project team' usually consists of only the project leader. This project leader has little authority over other employees and is involved in a light project that often consists of research work and giving advice to management. When such project leaders need the efforts of other employees in the organisation, they can either ask for help informally or arrange it through their supervisors.

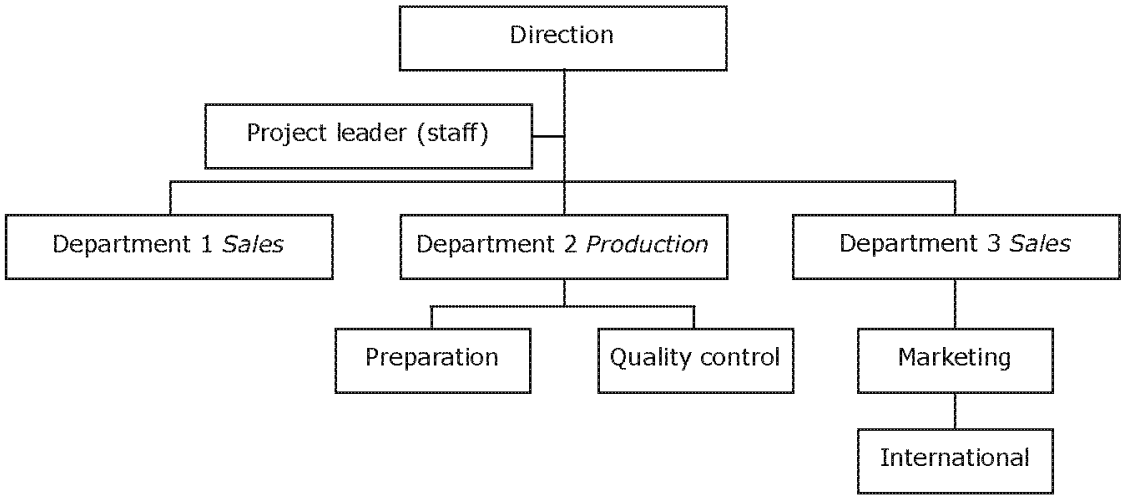


Figure 13: A project using the consultation structure. The position of the project leader (who is often a staff member) is completely outside the departments.

In the matrix structure, the organisation structure is arranged such that team members work (part-time) in project teams and in their positions in the line organisation (also part-time). It is also possible for matrix organisations to include people who are assigned to multiple projects simultaneously.

One major advantage over the consultation structure is that the matrix structure involves project teams that are comprised of multiple employees. This allows a project team to achieve more than it would in the consultation structure, in which the project leader must often go it alone. Project leaders have more authority within a matrix structure, as they actually supervise their own teams. This is likely to cause confusion among team members, who must then report to two supervisors: the project leader and the head of the department in which they work. Employees who are involved in more than one project simultaneously may have even more than two supervisors.

Department heads and project leader(s) should arrive at clear mutual agreements concerning the allocation of personnel. Experience has shown that this does not always occur. Employees are pulled from all sides, and they give first priority to the work of the supervisor who is screaming the loudest. This obviously does not always reflect the highest priority of the organisation as a whole.

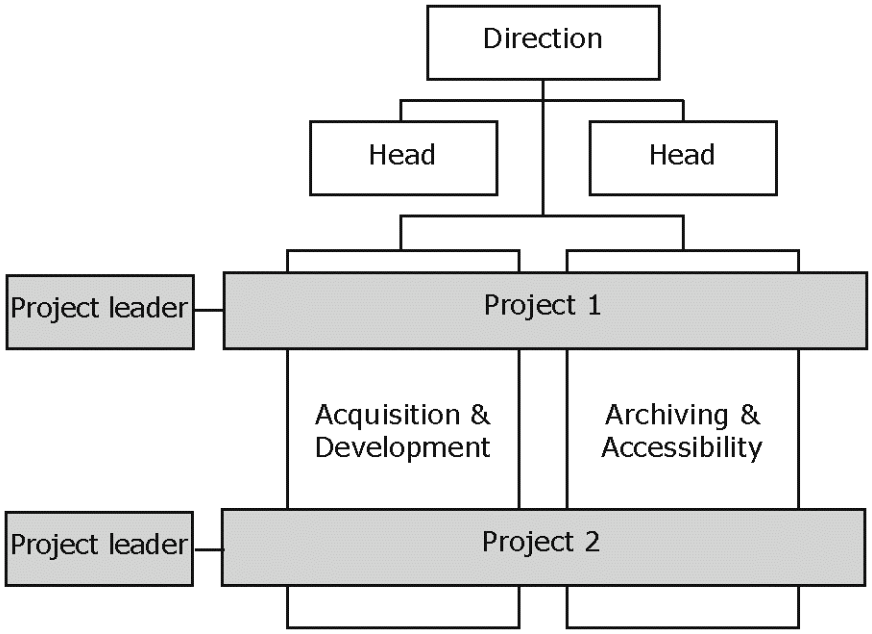


Figure 14: Projects organised according to the matrix model. Members of a project team have two supervisors: the project leader and the department head.

In the pure project structure, employees are taken out of the organisation to work solely on the project throughout its entire duration. This form is the most appropriate for large, heavy projects. The project team is largely autonomous and has only the project leader as a supervisor. One important disadvantage of this structure is that it is expensive and radical for the organisation, as employees are taken away from their original work for long periods.

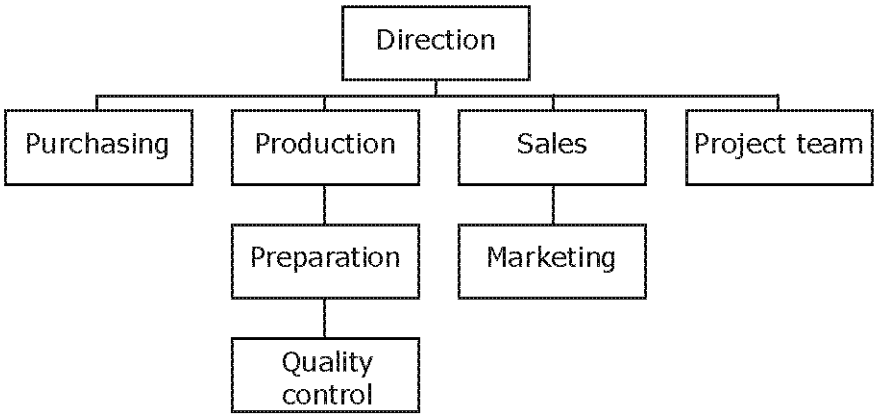


Figure 15: Diagram of pure project structure. The project team is in a relatively autonomous position outside the rest of the organisation.

Which structure is best depends primarily on the projects that are to be performed. The consultation structure is well suited for small projects, and the pure project structure is most appropriate for large, lengthy and heavy projects. In practice, many projects are arranged according to a matrix structure, because most projects fall somewhere between these two extremes. The coordination of projects is the most difficult in the matrix structure.

Appendix 1: Top 11 causes of delays in IT projects

This appendix contains a description of the eleven most common causes of delays in projects. For a detailed analysis of these and other causes of delays, see the works by McConnell and by Goldratt (McConnell, 1996, Goldratt, 2002).

1. Expansion of functionality
The expansion of functionality is a phenomenon in which new functionalities continue to be conceived and requested as the project proceeds. The software can never be completed in this way.

2. Gold plating
Gold plating is a phenomenon in which programmers and designers try to make many details of the software or design too elaborate. Much time is spent improving details, even though the improvements were not requested by the customer or client. The details often add little to the desired result.

3. Neglecting quality control
Time pressure can sometimes cause programmers or project teams to be tempted to skip testing. This frequently causes more delays than it prevents. The time that elapses before an error is discovered in the software is associated with an exponential increase in the time that is needed to repair it.

4. Overly optimistic schedules
Overly optimistic schedules place considerable pressure on the project team. The team will initially attempt to reach the (unrealistic) deadlines. These attempts lead to sloppy work and more errors, which cause further delays.

In this regard, be particularly wary of schedules that are imposed from above. The desire to complete a project (more) quickly sometimes arises for primarily strategic reasons; if it is not feasible, however, it should not be attempted. The project will not proceed more quickly and the product will ultimately suffer.

5. Working on too many projects at the same time
Dividing work across many different projects (or other tasks) causes waiting times that lead to many delays in projects.

6. Poor design
The absence (or poor realisation) of designs leads to delays, as it requires many revisions at later stages.

7. The 'one-solution-fits-all' syndrome
Using the right software for a project is important. Some software platforms are more suited to particular applications than others are. Thinking that the use of particular software will greatly improve productivity, however, is also a trap.

8. Research-oriented projects
Projects in which software must be made and research must be conducted are difficult to manage. Research is accompanied by high levels of uncertainty. When or if progress will be achieved in research is unclear. When software development is dependent upon the results of research, the former frequently comes to a standstill.

9. Mediocre personnel
Insufficiently qualified personnel can cause project delays. Technically substantive knowledge of the subject of the project plays a role, as do knowledge and skills in working together to play the game of the project.

10. Customers fail to fulfil agreements
Customers are not always aware that they are expected to make a considerable contribution to the realisation of a project. When customers do not react in a timely manner to areas in which they must be involved, projects can come to a standstill. Worse yet, the team may proceed further without consulting the customer, which can lead to later conflicts.

11. Tension between customers and developers
The tension that can arise between customers and developers (e.g. because the project is not proceeding quickly enough) can cause additional delays, as it disturbs the necessary base of trust and the working atmosphere.

Appendix 2: Roles within a project

This appendix provides definitions for the various roles of people who are part of a project.

1. Project members/Project team
The project members are the team members of the project – those who actually carry out the project and those who have tasks within the project. Team members often have differing areas of expertise. Team members can be internal (company personnel), external (from project partners, customers, users or temporary personnel) or both.

2. Project leader
The project leader is the one who directs the project team and has ultimate responsibility for the project result. Depending upon what has been agreed, a project leader can obviously delegate responsibility to team members, and external managers may be responsible for some components of the project.

In cyclical projects, the project leader represents the interests of both the customer and the programmers. Project leaders ensure that customers receive adequate technical explanation and help them to choose and prioritise functionalities.

3. Project manager
The terms 'project leader' and 'project manager' are often used interchangeably. A project manager is usually responsible for multiple projects, while a project leader usually has only one. Project leaders are thus located 'closer to the work floor' than are project managers, who are usually more involved with direction and numbers. Other meanings and definitions also exist, and the terms are often used interchangeably.

4. Programme manager
The programme manager is the one who evaluates a number of projects within an organisation. Project leaders and project managers report to the programme manager, who is often a member of the management team.

5. Customer
The customer is the entity that has ordered the project result. Customers may participate actively in the project or maintain a greater distance. Although customers are sometimes also the users of the project result, this is not always the case. Consider the example of a university that wants a web application for its employees and students. In this case, the university is the customer, and its employees and students are the users.

6. Users
Users are the people who will actually use the project result. It is important to involve the users in the definition phase, design phase and in the testing of the project result

7. Project partner

The project partner is a third party (organisation) with whom the project is conducted. If several parties are participating in the project, it is obviously important to define and delimit responsibilities clearly.

8. Client/customer/sponsor
The party or parties that make the project financially possible. In many cases, this is also the party that will use the project result. The client
ensures the financing for the project and is also the party to whom reports on the project result are directed.

Appendix 3: Useful resources for project management

1. http://sourceforge.net/
Website where open-source software can be found, including software for managing projects. The following open-source software can be downloaded here.

2. Xplanner
Xplanner is an open-source software tool for the administration and management of the cycles through story cards (according to the eXtreme Programming working method).

3. Open-source CVS (Current Version System) administrative applications that are frequently used include CVS, Subversion and Gnu arch.

4. MS Project, Fasttrack and others
MS project is the best-known programme for carrying out the administration of a project and for making Gantt charts (bar graphs).
Fasttrack is another well-known package, and there are many other open-source packages. These programmes are actually suitable only for projects that are conducted according to the waterfall method.

5. http://www.bugzilla.org/
Bugzilla is an open-source programme for the registration, protection and archiving of issues and bugs. This application is used primarily in software development.

Appendix 4: License for this handbook

Appendix 5: About DANS and the makers of this handbook

Data Archiving & Networked Services (DANS) is the national organisation in the Netherlands that provides for the storage of and perpetual access to data from research in the liberal arts and social sciences. To this end, DANS works together with researchers and encourages cooperation among scientists. DANS has the form of a network, with a centre that is responsible for the data infrastructure. This centre is comprised of a team of approximately fifteen people who work at the DANS office in The Hague or at one of the research centres throughout the country. For more information, please visit http://www.dans.knaw.nl

Author:
Wouter Baars develops software and education. He studied business administration at the Eindhoven University of Technology. Since completing his studies, he has worked on a variety of projects in the area of old and new media. He worked as the project leader for the Waag Society, KPN, the Digitale Universiteit, the Hogeschool van Amsterdam, Noterik Multimedia and the European Commission, among other entities. In addition to his work as a developer, Baars teaches in the area of project management. More information on his work is available on the following website: http://www.wouterbaars.net

Advisors:
Dr. Henk Harmsen is the adjunct director of DANS, which is a new initiative of the Royal Netherlands Academy of Arts and Sciences (KNAW) and the NWO in the area of archiving and accessibility of research data in the Netherlands. Harmsen studied computer applications in the humanities at the Universiteit van Amsterdam and received a PhD from the Vrije Universiteit van Amsterdam. His broad work experience includes positions as librarian, head of computerisation and head of business operations. More information about Harmsen is available on the following website:
http://www.dans.knaw.nl/nl/over_dans/organisatie/henk_harmsen/

Rutger Kramer studied Information Technology at Delft University of Technology. For his graduation project, Kramer was involved with the ECPA Sepia project, in which he collaborated with the Netherlands Institute for Scientific Information Services (NIWI - KNAW) on a meta-data entry application. After completing his internship, he remained with NIWI as a technical scientific programmer. In this position, he worked on a variety of projects, including EVAMP and XPAST, which focused on the disclosure of digital heritage materials. As an information scientist with DANS, Kramer serves as IT liaison and project manager for internal and external R&D projects. He is involved in the Easy Store DMS project for DANS, in addition to providing database disclosure for the Faculty of Letters at Utrecht University.
http://www.dans.knaw.nl/nl/over_dans/organisatie/rutger_kramer/

Laurents Sesink is an information scientist in the department of Acquisition and Development at DANS. Sesink studied history at Utrecht University and historical information technology at Leiden University. As a former senior digitalisation-services, technical scientific programmer, development-group co-ordinator, senior consultant/project leader and policy worker, Sesink has a broad background in the area of scientific and administrative information services.

http://www.dans.knaw.nl/nl/over_dans/organisatie/laurents_sesink/

Joris van Zundert is a researcher and developer with the Huygens Institute, which is a subsidiary of the Royal Netherlands Academy of Arts and Sciences (KNAW). He studied Dutch Language and Culture at Utrecht University. In addition to and following his studies, he developed a professional career as an independent designer and developer of Internet applications. He later combined education and practical experience while in service for the Netherlands Association for Science and Technology, the Netherlands Institute for Scientific Information Services and the Huygens Institute. In several projects, van Zundert has developed a variety of projects involving Internet applications and digital tools that are specifically focused on (literary historical) scientific use and research.
http://www.huygensinstituut.knaw.nl/index.php?option=com_content&task=view&id=131&Itemid=57&lang=du

Appendix 6: Sample action-and-decision list

Project name: <enter project name>
Date: <enter date of last changes>
Owner: <enter the name of the person who administers this document>
Phase: <enter one of the following: initiation phase, definition phase, design phase, development phase, implementation phase or follow-up phase>

Action list

#	Topic	Owner	Date planned	Completion date	Status
1	Enter the tasks that must be carried out within a given period	Name of the person who is responsible for this task	5-1-2006	3-1-2006	☺
2					☹
3					😐
4					

Decision list

#	Description	Date
1.	Enter descriptions of decisions that have been taken in consultation	Date of the decision
2.		
3.		

Appendix 7: Sample Issue log

Project name: <enter project name>
Date: <enter date of last changes>
Owner: <enter name of the person who administers this document>

Nr.	Type	Issue description	Name	Date	Priority	Decision	Status
1.	RFC	Enter a brief description of the issue that arose			1= high 3= low	Describe the decision here	ok
	DS					T = accepted	
	Q					A = rejected	
	C					U = postponed until...	
	R						

Type	Priority	Decision	Status
RFC= Request for change (general)	1 = Immediate action	A = Accepted	OK = Issue has been resolved
DS = Deviation from specifications (with regard to design)	2 = Take action later	R = Rejected	Open = Awaiting resolution
Q = Question	3 = No action	P = Postponed until...(Date/event)	
C = Concern			
R = Risk			

Appendix 8: Sample Risk log

Project name: <Enter project name>
Date: <enter date of last changes>
Owner: <enter name of the person who administers this document>

#	Description of risk	Priority	Measure	Status
1.	Enter a brief description of the perceived risk	1= high 3= low	Describe the measures that were taken	ok

Priority	Status
1 = take immediate action	OK = Risk has been resolved.
2 = take action later	open = awaiting action
3 = take no action	

Appendix 9: Sample meeting report

Minutes of Project: <Enter project name>
Date: <enter the date of the conversation/meeting>
Note-taker: <enter the name of the person who prepared this document>
Present: <enter the names of those who were present for this discussion>
Absent with notice: <enter the names of those who were absent with notice
Absent without notice: <enter the names of those who were absent without notice>

Agenda

<Enter the list of topics as they appeared on the agenda during the meeting. For example:>

Approval of the minutes from the previous meeting
Discussion of action list
Discussion of decision list
Discussion of issue log
Discussion of risk log (new risks and obstacles)
Project progress
Schedule adjustments
Consultation with management
Question round

1. Approval of minutes from previous meeting
Piet remarked that his comments concerning the developments of new software by the competition were not accurately recorded in the report. He will send a brief e-mail to the note-taker with a correct reflection of his ideas. The others in attendance approved the minutes.

2. Action list
A number of actions were cancelled, and other new actions were reported.
<<Please refer to the separate document entitled ActionsAndDecisions.doc>>

3. Decision list
Please refer to the separate action-and-decision list for information on the decisions that were taken.
<<Please refer to the separate document entitled ActionsAndDecisions.doc>>

4. Issue log
Please refer to the separate issue log for information on the issues that are currently awaiting action.
No new issues were reported this week.

5. Risk log
Henk has learned of a new risk that we had overlooked. What should we do if our software supplier declares bankruptcy? This risk has been noted in the risk log. Klaas will consider the matter further and see what our contracts with the supplier say in this regard.
Please refer to the risk log for further information.

6. Project progress

Partner 1
The Java wing is working hard to realise the software. Three engineers are now assigned to the project.
The testers are working to prepare the test scripts. Henk asked Marie whether he had sent this to Floor. Kees reported that the plans for detail testing are not deliverables. The test scripts are, however, if the partner wishes insight into the plans in question. This is obviously always possible.

Partner 2
Floor reported that she is now working on a new name for the product. Henk will go ahead and prepare a change proposal that will explain the impact of this change on the project.
The estimate for licenses has now been received from the supplier.
Floor remarked that she would like to submit another report to the financer in early October. We agreed to submit the report to the partner five working days after the end of September.

Participants in the previous period

Name	Position
Piet Pieterse	Operations manager
Jan Jansen	Project leader
Etc.	Lead Engineer Client PRODUCT
	Technical Architect
	Lead Engineer Server PRODUCT
	Java Engineer
	Java Engineer
	Quality/Testing manager
	Testing Coordinator

Participants in the coming period

Name	Position
Klaas Klaaszoon	Operations manager
Marie de Boer	Project leader
Etc.	Lead Engineer Client PRODUCT
	Technical Architect
	Lead Engineer Server PRODUCT
	Java Engineer
	Java Engineer
	Java Engineer
	Java Engineer
	Java Engineer
	Java Engineer
	Quality/Testing manager
	Testing Coordinator
	Tester

7. Schedule adjustments

The chart below reflects the new schedule, which the current project partners consider realistic at this time.

Phase/Milestone	Starting date/Milestone	Ending date	Who
Preparation	7-4-03	6-6-2003	Partner1
Design	7-4-03	6-6-2003	Partner1 + Partner2
Decision-making	10-6-2003	13-6-2003	Partner1 + Partner2
Design approval	13-6-2003	13-6-2003	Partner2
Implementation/Testing	30-6-2003	28-11-2003	Partner1
Delivery of first version	28-11-2003	28-11-2003	Partner1
Test of acceptance	1-12-2003	2-01-2004	Partner2
Support for acceptance test	1-12-2003	2-01-2004	Partner1
Acceptance	2-01-2004	2-01-2004	Partner2
Substantive user test	2-01-2004	25-06-2003	Partner2
Support for substantive user test	2-01-2004	25-06-2003	Partner1
Optimisation	28-06-2003	27-08-2003	Partner1
Delivery of second version	27-08-2003	27-08-2003	Partner1
Guarantee	27-08-2003	26-11-2004	Partner1

8. Consultation with management
In his role as project leader, Klaas had consulted with management. Management would like the project to be completed within one month, or two at the most. Klaas reported that he did not consider this feasible, but that the team would do its best to make it (the impossible) possible.

9. Question round
Marie remarked that the meeting had already lasted for two hours, even though the goal had been to limit it to forty-five minutes. She asked everyone to try to keep the meetings short.

Henk states that he will be unable to attend the next meeting.

Next meeting(s)

Type:	Progress conference
Frequency:	weekly
Day:	Tuesday 19 Augustus 2006
Time:	13:00
Location:	Partner 1, Rotterdam office
Attendees:	Klaas, Henk, Floor, Marie
Absent:	Henk

Tentative agenda:
1. Meeting report from previous meeting
2. Action list
3. Decision list
4. Deviations from specifications
5. Project progress
6. Scheduling (milestones/changes)
7. Overwork/underwork
8. Issue log
9. Risks and obstacles
10. Other matters/Question round

Appendix (appendices)

Action/Decision list for the Project
Issue log
Risk log

Appendix 10: Sample project plan

<project name>

Project name: <enter project name >
Date: <enter date of last changes>
Project leader: <enter name(s) of project leader(s)>
Phase: <enter one of the following: initiation phase, definition phase, design phase, development phase, implementation phase or follow-up phase>

For approval: <name + signature of project leader>

Date:

For approval: <name + signature of client>

Date:

Introduction:

This document is a brief instruction book for compiling a project plan. The first project plan is developed after the initiation phase and serves as an approval document for the entire project. After each phase, this document must be revised. A detailed plan must be developed for the next phase. For the subsequent phases, the plan will be of a more global nature. Evaluation of this document also takes place after each phase.

General information about the project

1. Situation sketch and problem definition of the project

- Provide a brief description of the organisation in which the project will take place.
- Provide a brief description of the department(s) in which the project will take place.
- Provide a brief description of any relationships between this project and any others.
- Provide a brief description of the history of this project.
- Provide a brief description of the catalyst for this project.
- Identify the client(s) for this project.
- Identify the contractor for this project.

2. Project assignment

- Explain the rationale for this project; follow the SMART formula as closely as possible.
  (SMART=specific, measurable, attainable, realistic, timely).
- Identify what will be delivered.
- Identify the most important boundaries of the project (what will not happen).

3. Risk analysis

- Identify the risks that are known in this project.
- Specify how these risks will be handled (avoiding, fighting, insuring, accepting).

4. Organisation of the project

This section of the project plan provides a description of the phases, activities in each phase and the associated control factors of the project. The plan is elaborated globally for the more remote phases and in concrete and specific terms for the next phase. This sample begins from the compilation of the initial project plan and thus with the initiation phase. This section of the project plan should be revised after *each* phase.

4.1 Explanation of the project/management model that will be applied

This project plan is based on the waterfall method/DANS method <select one of the models here>, which is described in the DANS Handbook for Project Management. The handbook is included with this project plan.

4.1.1 Initiation phase
This project plan is the result of the initiation phase. This phase requires no further detailed elaboration. A summary of the activities that have taken place in preparation may be included.

4.1.2 Definition phase
Planned starting date: <date>
Planned ending date: <date>

Description of the result of the definition phase:

A list of requirements concerning the project result will be compiled in the definition phase.

Most important milestones: <(example)>

- List of functional requirements
- Research on legal requirements
- Requirements from interviews with end users
- Requirements from end-user tests
- Report of technical requirements
- Client approval of list of requirements

<Specify when each milestone will be accomplished and who is responsible. Specify the required quality for each milestone (intermediate product).>

Activities in the definition phase:
Provide a list of the activities that must take place in order to achieve the milestones. Specify who will carry out these activities, when and by whom they will be approved (ultimate responsibility).

Timeline:
Include a chronological list of activities using a bar graph or similar visual aid. Provide a clear indication of the timing of all milestones. Do not forget to include margins.

Budget:
Provide an estimate of the costs for each activity in this phase. In addition, specify the costs for materials and supplies, as well as any other costs. Refer as necessary to external documents that specify the costs (see also the separate model for budgets).

Internal information:
Indicate how the information from this phase will be recorded and archived. Specify any resources that will be used and, if necessary, who will or will not have access to this information.

External information:
The approval of this phase by the client/customer is an important information moment. Indicate the reports that must be submitted to the client, customer or external management after this phase.

4.1.3 Design phase
Planned starting date: <date>
Planned ending date: <date>
Description of the result of the design phase:
A (number of) design(s) for the intended project result will be made in the design phase.

Most important milestones: <(example)>
One or more dummies
Screen design
Photo impressions/sketches

<Specify when each milestone will be accomplished and who is responsible. Specify the required quality for each milestone (intermediate product).>

Activities in the design phase:
Provide a list of the activities that must take place in order to achieve the milestones. Specify who will carry out these activities, when and by whom they will be approved (ultimate responsibility).

Timeline:
Include a chronological list of activities using a bar graph or similar visual aid. Provide a clear indication of the timing of all milestones. Do not forget to include margins.

Cost estimates:
Provide an estimate of the costs for each activity in this phase. In addition, specify the costs for materials and supplies, as well as any other costs. Provide an estimate of the costs for each activity in this phase. As necessary, refer to an external document that contains the costs. Do not forget to include a category of 'unexpected costs' and the costs of project management itself (also see the separate model for budgets). As necessary, refer to an external document that contains the costs (also see the separate model for budgets).

Internal information:
Indicate how the information from this phase will be recorded and archived. Specify any resources that will be used and, if necessary, who will or will not have access to this information.

External information:
The approval of this phase by the client/customer is an important information moment. Indicate the reports that must be submitted to the client, customer or external management after this phase.

After the design phase, the waterfall method continues with the development phase; the DANS method for software development continues with the cyclical section. The various possibilities are described together below. One of the two options must be chosen for the preparation of a project plan.

4.1.4 Development phase <(Waterfall only)>
Planned starting date: <date>
Planned ending date: <date>
Description of the result of the development phase:
During the development phase, an action plan will be developed in preparation for the implementation phase. <Note: This phase is not always necessary for every project. Particularly for smaller projects, this phase may be omitted.>

Most important milestones: <(example)>
Action plan, Part 1
Action plan, Part 2
etc.

<Specify when each milestone will be accomplished and who is responsible. Specify the required quality for each milestone (intermediate product).>

Activities in the development phase:
Provide a list of the activities that must take place in order to achieve the milestones. Specify who will carry out these activities, when and by whom they will be approved (ultimate responsibility).

Timeline:
Include a chronological list of activities using a bar graph or similar visual aid. Provide a clear indication of the timing of all milestones. Do not forget to include margins.

Cost estimates:
Provide an estimate of the costs for each activity in this phase. As necessary, refer to an external document that contains the costs. Do not forget to include a

category of 'unexpected costs' and the costs of project management itself (also see the separate model for budgets).

Internal information:
Indicate how the information from this phase will be recorded and archived. Specify any resources that will be used and, if necessary, who will or will not have access to this information.

External information:
The approval of this phase by the client/customer is an important information moment. Indicate the reports that must be submitted to the client, customer or external management after this phase.

4.1.5 Implementation phase <(waterfall only)>
Planned starting date: <date>
Planned ending date: <date>
Description of the result of the implementation phase:
The project result will be built in the implementation phase.

Most important milestones: <(example)>
Element 1 of the implementation
Element 2 of the implementation
etc.

<Specify when each milestone will be accomplished and who is responsible. Specify the required quality for each milestone (intermediate product).>

Activities in the implementation phase:
Provide a list of the activities that must take place in order to achieve the milestones. Specify who will carry out these activities, when and by whom they will be approved (ultimate responsibility).

Timeline:
Include a chronological list of activities using a bar graph or similar visual aid. Provide a clear indication of the timing of all milestones. Do not forget to include margins.

Cost estimates:
Provide an estimate of the costs for each activity in this phase. In addition, specify the costs for materials and supplies, as well as any other costs. As necessary, refer to an external document that contains the costs (also see the separate model for budgets).

Internal information:
Indicate how the information from this phase will be recorded and archived. Specify any resources that will be used and, if necessary, who will or will not have access to this information.

External information:
The approval of this phase by the client/customer is an important information moment. Indicate the reports that must be submitted to the client, customer or external management after this phase.

4.1.6 Cyclical phase <(only the DANS method for software development)>
Planned starting date: <date>
Planned ending date: <date>
Description of the result of the implementation phase:
During the cyclical phase, the project result will be further examined, specified and built.

Number of hours available for the cyclical phase:
Indicate how many hours are available for the cyclical phase, possibly distributed over the various team members, if the distribution is not equal.

Preliminary estimate of the number of cycles and their products <(example)>
Cycle 1: basic architecture
Cycle 2: interaction between servers
Cycle 3: interaction with customer
etc.

This section is important primarily for compiling the initial project plan. As the cyclical phase approaches, the planning system shifts to the use of story cards (see DANS Handbook for Project Management).

Participants in the cyclical phase:
Provide a list of participants in the cyclical phase and their responsibilities:
<example>

Jan Jansen: Programmer
Piet Pietersen: Programmer
Marie Pedro Del Mar: Programmer
Kees Keeszoon: Designer (interaction and graphics)
Client: Process information and testing
etc.

Cost estimates:
Provide an estimate of the costs for each activity in this phase. In addition, specify the costs for materials and supplies, as well as any other costs. As necessary, refer to an external document that contains the costs (also see the separate model for budgets).

Internal information:
Indicate how the information from this phase will be recorded and archived. Specify any resources that will be used and, if necessary, who will or will not have access to this information. <Story cards will be used, possibly supplemented with a project log, CVS system, bug tracker and tools (e.g. Xplanner>.

External information:
The approval of this phase by the client/customer is an important information moment. Indicate the reports that must be submitted to the client, customer or external management after this phase.
4.1.7 Follow-up phase<(waterfall and DANS method for software development)>
Planned starting date: <date>
Planned ending date: <date>

Note: Provide a clear indication of when evaluation will end in the project team.

Description of the result of the evaluation phase:
The project will be completed during the follow-up phase. Indicate what is and is not included in completion.

Most important milestones: <(example)>
Project report
Transfer document for administrative organisation
Project website with user information
etc.

<Specify when each milestone will be accomplished and who is responsible. Specify the required quality for each milestone (intermediate product).>

Activities in the follow-up phase:
Provide a list of the activities that must take place in order to achieve the milestones. Specify who will carry out these activities, when and by whom they will be approved (ultimate responsibility).

Timeline:
Include a chronological list of activities using a bar graph or similar visual aid. Provide a clear indication of the timing of all milestones. Do not forget to include margins.

Cost estimates:
Provide an estimate of the costs for each activity in this phase. In addition, specify the costs for materials and supplies, as well as any other costs. Refer as necessary to external documents that specify the costs (also see the separate model for budgets).

Internal information:
Indicate how the information from this phase will be recorded and archived. Specify any resources that will be used and, if necessary, who will or will not have access to this information.

External information:
The approval of this phase by the client/customer is an important information moment. Indicate the reports that must be submitted to the client, customer or external management after this phase.

Overview

The final section of the project plan provides an overview of the costs and the timeline for the entire project.

Phase	Expected starting date	Expected date of completion	Total cost estimate
Definition			
Design			
Preparation			
Implementation			
Follow-up			
		Total:	

Appendix 11: Sample budget

Budget
<Project name>
<Project phase: initiation - evaluation >
<Name of the person compiling this document >
<date>

Amount in Euros

Initiation phase	*Who?*	*Hours*	*Fee*	*Total*
Research draft	Project Leader	20	75	1,500
Consultation with partners	Project Leader	10	75	750
Preparation of project proposal	Department Head	40	80	3,200
Proposal design	Graphic Design	10	50	500
Contract expertise			1,000	1,000
...				
			subtotal initiation phase:	6950
Definition phase				
Consultation with end users	Project Leader	10	75	750
	Functional Designer	10	65	650
Consultation with experts	Project Leader	20	75	1,500
	Functional Designer	30	65	1,950
	Internal Experts	30	100	3,000
Contract expertise				2,500
Catering for meetings				250
Project management		25	75	250
			subtotal definition phase:	10,850
Design phase				
Travel costs	Team Members			1,000
Design sketch	Designer	60	60	3,600
Design presentation	Designer	20	60	1,200
Materials				2,000
Web presentation				
Content	Copy writer	20	60	1,200
HTML	Programmer	20	65	1,300
Design	Designer	30	60	1,800
Project management		15	75	1,125
			subtotal design phase:	13,225

Preparation phase

Travel costs				2,000
Material costs				15,000
Elaboration of schedules	Project Leader	40	75	3,000
Project management		15	75	1,125
		subtotal preparation phase:		22,125

Implementation phase

Travel costs (personnel)				2,000
Material transport costs				3,000
Operations (internal personnel)	Carpenters	160	45	7,200
	Plumbers	160	50	8,000
	Designers	80	65	5,200
Temporary workers				25,000
Gas, water, electricity				3,000
Project management		80	75	6,000
		subtotal implementation phase:		59,400

Follow-up phase

Travel costs (personnel)				2,000
Material disposal				3,000
Material clean-up	project team	40	60	2,400
Final report	Project Leader	40	75	3,000
Accountant's statement				1,000
Project management		40	75	3,000
		subtotal follow-up:		15,900

Summary

subtotal initiation phase:		6,950	
subtotal definition phase:		10,850	
subtotal design phase:		13,225	
subtotal preparation phase:		22,125	
subtotal implementation phase:		59,400	
subtotal follow-up:		15,900	
	subtotal:		128,450
unexpected costs 10%			13,495

total:	141,945

This budget is a global estimate of the entire <project name> project. At the beginning of each new phase, a detailed budget for that phase is delivered.

Appendix 12: Sample financial statement

Financial report
<Project name>
<Name of the person compiling this document>
<date>

		Amount in Euros				
Initiation phase	*Who?*	*Budgeted hours*	*Actual hours*	*Fee*	*Total budgeted*	*Actual total expenditures*
Research draft	Project Leader	20	23	75	1,500	1,725
Consultation with partners	Project Leader	10	12	75	750	900
Preparation of project proposal	Department Head	40	35	80	3,200	2,800
Proposal design	Graphic Design	10	20	50	500	1,000
Contract expertise				1000	1,000	980
...						
Unexpected costs						861
		subtotal initiation phase:			6,950	7,405
Definition phase						
Consultation with end users	Project Leader	10	10	75	750	750
	Functional designer	10	10	65	650	650
Consultation with experts	Project Leader	20	18	75	1,500	1,350
	Functional designer	30	35	65	1,950	2,275
	Internal experts	30	40	100	3,000	4,000
Contract expertise					2,500	2,700
Catering for meetings					250	247
Project management		25	41	75	250	3,075
Unexpected costs						120
		subtotal definition phase:			10,850	15,167
Design phase						
Travel costs	Team members				1,000	1,221
Design sketch	Designer	60	63	60	3,600	3,780
Design presentation	Designer	20	19	60	1,200	1,140
Materials					2,000	1,873
Web presentation						
Content	Copy writer	20	18	60	1,200	1,080
Html	Programmer	20	32	65	1,300	2,080
Design	Designer	30	18	60	1,800	1,080
Project management		15	12	75	1,125	900
Unexpected costs						1,245
		subtotal design phase:			13,225	14,399

Development phase

Travel costs					2,000	1,200
Material costs					15,000	13,981
Elaborating schedules	Project Leader	40	45	75	3,000	3,375
Project management		15	22	75	1,125	1,650
Unexpected costs						11,003
		subtotal development phase:			22,125	31,209

Implementation phase

Travel costs (personnel)					2,000	2,100
Costs for material transport					3,000	2,850
Operations (internal personnel)	carpenters	160	200	45	7,200	9,000
	plumbers	160	165	50	8,000	8,250
	designers	80	75	65	5,200	4,875
Temporary workers					25,000	24,560
Gas, water, electricity					3,000	3,800
Project management		80	81	75	6,000	1,650
Unexpected costs						231
		subtotal implementation phase:			59,400	57,316

Follow-up phase

Travel costs (personnel)					2,000	1,642
Material disposal					3,000	3,500
Material clean-up	Project team	40	45	60	2,400	2,700
Final report	Project Leader	40	40	75	3,000	3,000
Accountant's statement					1,000	981
Project management		40	62	75	3,000	4,650
Unexpected costs						861
		subtotal follow-up:			15,900	17,334

Summary

	budget	
subtotal initiation phase:	6,950	7,405
subtotal definition phase:	10,850	15,167
subtotal design phase:	13,225	14,399
subtotal development phase:	22,125	31,209
subtotal implementation phase:	59,400	57,316
subtotal follow-up:	15,900	17,334
unexpected costs 10%	13,495	14,321
total:	141,945	157,151

This financial report should be accompanied by a brief analysis of the largest differences between the budget estimate and actual costs.

Literature

Chromatic and Diaz, T. Apandi (Ed.) (2003). *Extreme Programming Pocket Guide.* Sebastopol, CA: O'Reilly & Associates.

Goldratt, E.M. (2002). *De zwakste schakel.* Utrecht: Het Spectrum.

Kor, R. and Wijnen, G. (2002). 50 checklisten voor project- en programmamanagement. Deventer: Kluwer.

Kroll, P. and Kruchten, Ph. (2004). The Rational Unified Process Made Easy: A practitioner's Guide to the RUP. Boston: Addison Wesley.

McConnell, S. (1996). *Rapid Development: Taming Wild Software Schedules.* Redmond, Washington: Microsoft Press.

Stapleton, J. (2002). DSDM Dynamic Systems Development Method: De methode in de praktijk. Schoonhoven: Academic Service.

Wijnen, G., Renes, W. and Storm, P. (2004). *Projectmatig werken.* Utrecht: Het Spectrum.

Internet sources

[i] Spolsky, J. (2000). Painless Functional Specifications - Part 1: Why Bother? Referred to 31 May, 2006 via http://www.joelonsoftware.com/articles/fog0000000036.html

[ii] Why power your site with XP? (z.j.). Referred to 31 May, 2006 via http://www.XP.com/

[iii] Extreme Programming: A gentle introduction (Last modified February 17, 2006). Referred to 31 May, 2006 via http://www.extremeprogramming.org/

[iv] Spolsky, J. (2000). WhatTimeIsIt.com. Functional Specification. Referred to 31 May, 2006 via http://www.joelonsoftware.com/articles/WhatTimeIsIt.html

[v] Spolsky, J. (2002). Fire And Motion. Referred to 31 May, 2006 via http://www.joelonsoftware.com/articles/fog0000000339.html

[vi] Pair Programming successes and failures (2001-2003). Referred to 31 May, 2006 via http://discuss.fogcreek.com/joelonsoftware/default.asp?cmd=show&ixPost=2058