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.

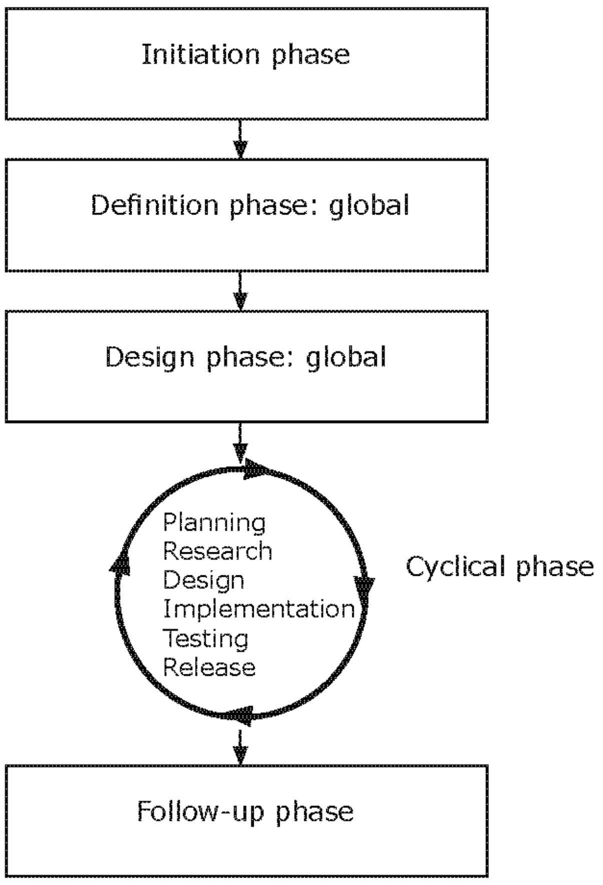

Figure 11: Schematic illustration of the DANS software-development method

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.

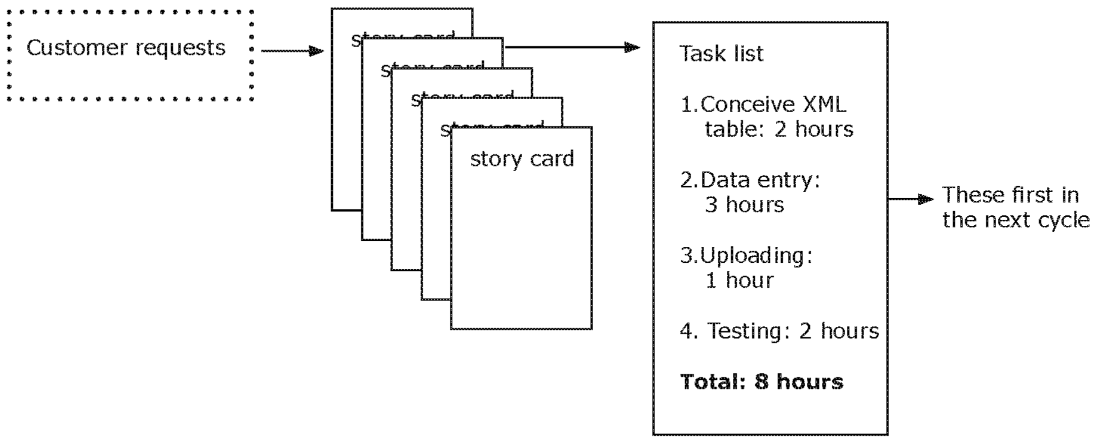

Figure 12: Planning functionalities in the cycles

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