CHAPTER VI
DESIGN OF NETWORK MANAGEMENT FUNCTIONS
Design of Network Management functions

Design of management functions

This chapter discusses the design of management functions. The objective of this
discussion is to show that management functions can be developed together with
primary functions as part of the same design process. For this purpose it is assumed that
the design is performed in a cyclic fashion, which implies that a 'cyclic design model'
can be applied.

The structure of this chapter is as follows. Section 6.1 gives a short introduction to the
cyclic design model. Section 6.2 through Section 6.4 apply this model to explain when
service, protocol and element management functions should be elaborated during such
design. Since the design of element management functions is the easiest to explain, it
will be discussed first. Service management will be discussed last.

In Section 6.5 the case is treated that during the development of management functions
additional management problems may appear. To resolve these problems it may be
necessary to introduce additional management functions that manage the initial
management functions.

Section 6.6 illustrates that it may not always be possible to complete the definition of all
management functions during the design phase. In several cases it may be possible,
however, to reconsider the unresolved management problems as part of the design of
future generations of network systems. To include the design of these management
functions, the cyclic design model will be extended to incorporate the development of
such future generations. In fact, the model that emerges may be considered as a
combination of the life cycle model and the cyclic design model.

Section 6.7 provides the conclusions.

6.1 Cyclic design

To provide a common basis for the remainder of this chapter, a short introduction to the
cyclic design model will be given[Schot J.: "The role of Architectural Semantics in the
formal approach of Distributed Systems Design", Ph.D. thesis, University of Twente,
1990].

The idea of cyclic design is to distribute design problems over a sequence of design
cycles, allowing to exploit design experience obtained in previous cycles while
performing the next.
Figure 6.1 shows an example of a cyclic design process. In this example, three design cycles are performed in sequence. During the first design cycle only a subset of the user requirements will be considered. During the second design cycle additional requirements will be considered, and during the last design cycle all requirements will be taken into account. The experience that is Design of management functions obtained in each of these cycles, is used as input to the subsequent cycles. In this example, each cycle comprises all phases of the step-wise design process: architectural, implementation and realization phase. The realization that satisfies all user requirements, will only be available after the last design cycle is completed; the earlier design cycles did not cover all user requirements and produced what may be called 'prototypes' [Pressman 1991].

A good strategy in cyclic design is to start with those requirements that are expected to have the strongest impact on the system's structure. Such strategy ensures that most of what has been achieved in the first cycle(s), can be reused in later cycles. Usually this
means that the designer should start with the development of primary, i.e. data transfer functions; development of management functions should be postponed until later cycles.

Example: In the first cycle the designer elaborates how to exchange data between systems. To reduce the complexity of this cycle, the designer neglects for instance QoS requirements. After completion of the first prototype, the designer detects that the prototype is unable to deliver all data: data sometimes get lost. In the next cycle the designer therefore adds retransmission functions. Although the new prototype behaves better, some data still get lost. The designer therefore adds in subsequent cycles fault management functions, which should be used to detect and possibly correct persistent faults.

At first glance, cyclic design looks inefficient with respect to design time and manpower. Two remarks should therefore be made:

- It is not necessary to consider all three phases of the step-wise design process during each design cycle. This is particularly true for the first cycle(s), in which the designer will primarily be interested in getting rough insight in the system's structure. Realization phases cost a lot of time, but do not contribute very much to this insight. It may therefore be a good idea to skip some of these realization phases.

Figure 6.2: Parallel activities in Cyclic design
Cyclic design allows several design groups to proceed in parallel. The following scenario is for instance conceivable (Figure 6.2). An architectural group develops the initial service definition. After its completion, an implementation group uses this definition to develop the initial protocol definition. In the mean time the architectural group starts to develop a revised service. After completion of the initial protocol, a realization group uses this definition to develop the first prototype. In the mean time, the implementation group continues and develops a revised protocol. The experience obtained in the first implementation cycle will also be used by the architectural group for the development of a better service, etc.

6.2 Development of element management

As explained in Section 5.4, the need for element management functions appears during the last phase of the step-wise design process: the realization phase. In this section this phase will be discussed separate from the architectural and implementation phase. The starting point for this phase is the protocol specification. In this section it is assumed that the cyclic design model will be applied to transform this protocol specification into a realization.

Consider the example of a protocol specification that includes flow control and retransmission functions. Assume the protocol should be realized in software. During the first cycle, the designer's main problem is to find a good realization structure. For this purpose, the designer considers the most important protocol functions first, in this example the flow control and retransmission functions (Figure 6.3). Although buffers will be needed for both functions, the designer need not worry during this first cycle about the amount of memory needed for these buffers. Just to be sure, the designer may decide to allocate a large amount of memory.

After completion of the first prototype, the designer may discover that these buffers account for 50% of all memory capacity. To reduce this percentage, the designer may decide to add a function that allows a manager to determine the required amount of memory during the start up. In case the system is expected to handle many messages (e.g. routers and file servers), the manager allocates large amounts of memory; in other cases (e.g. End Systems) the manager allocates small amounts of memory.

As discussed in Section 5.4, this function may be regarded as an element management function. Such function will be identified during the design of the primary functions and should be elaborated during a next cycle of the design. A problem with the type of element management as described above, is that memory allocation is static: the amount of memory allocated during system's start-up can not be changed after the system has become operational. In a next cycle the designer may therefore decide to add the possibility to allocate during the operational phase memory dynamically (Figure 6.3).
This type of management will be particularly useful in combination with a facility to monitor the actual memory usage. Such management function should therefore be added too. As a result, the element manager becomes able to fine-tune memory allocation based on operational requirements. A final step may be to include a possibility to monitor memory usage and allocate memory from a remote place (Figure 6.3), which involves the definition of special element management interactions. Since these management interactions change the external behaviour of the various systems, the initial protocol is thus extended.

The example in this section showed that element management functions can be identified during the first design cycle(s) as part of the realization process of the primary functions, and can be elaborated during the subsequent design cycles (Figure 6.4). To allow the initiation of element management functions from a remote place, it may be necessary to extend the initial protocol specification.

This implies that the introduction of element management functions can have repercussions for later implementation cycles. The need to introduce element management functions can usually not be deduced from the service definition: it will
therefore not be possible to address element management functions during the architectural phase.

6.3 Development of protocol management

The question when to elaborate protocol management functions will be discussed now. The scope of this discussion will be restricted to those management functions that can be completely defined before the operational phase starts; protocol functions elaborated after the start of the operational phase will be discussed in Section 6.6.

Protocol management functions should be elaborated during the implementation phase, and not in the architectural or realization phase. This can be understood as follows:

- To define protocol management functions, knowledge must be available about the primary protocol functions for which the management functions are intended. [Steege J.W. ter: "Alternatives for the OSI Systems Management Framework", 1991] Such knowledge about protocols is not available in the architectural phase, which implies that protocol management functions can not be defined during the architectural phase.

- To perform protocol management functions, it will generally be necessary to exchange management information between protocol systems. To allow systems
from different vendors to interoperate, all information exchanges must be
defined during the implementation phase. This implies that also protocol
management functions should be elaborated in the implementation phase, and
not in the realization phase.

A sensible implementation approach is to determine the main structure of the provider
first. In case the design is performed in a cyclic fashion, this implies that the primary
protocol functions should be defined during the initial design cycle(s). While elaborating
these functions, protocol management functions may be identified (see also Section 5.3).
To reduce the complexity of the design, the elaboration of protocol management
functions may be postponed until later design cycles.

This process is shown in Figure 6.5. In this figure the first cycle is used for the
development of an initial protocol definition, which includes functions that can directly
be derived from the service definition (such as those to exchange user data). Although
the need for protocol management functions is already apparent during this first cycle, such functions will not be elaborated during this first cycle. Instead, protocol management functions are gradually added within the subsequent design cycles. The result of each of these cycles is a 'richer' protocol.

6.4 Development of service management

During the start of the design the emphasis will in many cases be on user data aspects and not on management. This implies that the first questions the designer will be faced with, are questions like:

- Should the provided service be connection-oriented or connectionless?
- Should the network support a fixed or variable data size?
- Should the network support a fixed or variable delay?

It is likely that the answers to these questions have a major impact upon the network's internal structure.

Example: To a large extent, the choice between a circuit switched network with out-of-band signalling and a packet switched network with inband signalling will be determined by the answers to the above questions.

In case the design is performed in a cyclic way, it may be a good approach to elaborate user data issues during the first design cycle(s) and service management issues during later cycles (Figure 6.6).

![Diagram](image)

Figure 6.6: Cyclic Development of OSI Network Layer Protocol

The first cycle(s) will be used by the designer to transform the requirements concerning the exchange of user data into a service definition. This definition specifies the service primitives that can be used to exchange user data, the relationship between these service primitives and the parameters that belong to these primitives. An example of such service definition is the standard for the OSI network service. This definition does not
yet include service management issues, such as how to assign addresses or closed user groups (see also Section 5.2). These issues are not expected to determine the network's main structure and can therefore be elaborated during subsequent cycles.

Figure 6.7 shows that new design cycles do not only consider service management issues that can be derived from the user requirements, but also issues that came up during previous realization and implementation exercises. From this form of feedback the designer for instance learns which system failures are most likely to occur. This knowledge can be used in new design cycles to Development of service management decide which fault information should be presented to the service users and which fault management functions should be added.

Figure 6.7 : Service Management and Cyclic Design

The sequence in which service management issues will be addressed, is decided by the designer. The designer usually starts with those requirements that are expected to have the largest impact on the network's structure. It may be expected that service management issues that directly follow from the user requirements (such as the assignment of addresses and closed user groups) will be tackled before service management issues that can only be identified after some implementation or realization experience has been obtained (such as fault management).
In real life designs, service management functions will not always be addressed during the architectural phase. Instead, elaboration is sometimes postponed until the end of the realization phase. This may be a bad policy, as can be demonstrated by the following example.

Example: In real life designs, the problem of SAP address assignment is sometimes tackled as tail piece of the design. In such cases, address assignment functions are added by the various manufacturers just before the realization phase completes. As a result, address assignment will be system dependent and management interactions between systems will not be defined. Without such interactions, the network cannot check for duplicated addresses. Since uniqueness of SAP addresses must still be guaranteed, the burden is put upon the (human) manager, who must take special measures during the operational phase.

In case the problem of address assignment would have been resolved during the architectural phase, a special service primitive could have been defined to set or modify SAP addresses. This service primitive could be defined in such a way, that successful execution would only be possible in case the proposed address was unique. To support this primitive, the implementer should define protocol functions that check uniqueness of the proposed address. Since these functions are added during the design phase, special measures will not be needed during the operational phase.

6.5 Meta-management

Section 6.2 through Section 6.4 discussed the development of management functions. In a number of cases the complexity of these management functions will be such, that these functions themselves should also be managed. To denote this 'management of management', we introduce the term 'meta-management. Which meta-management
functions should be added can only be determined during the elaboration of the 'normal' 
management functions. Meta-management functions must therefore be elaborated in 
later design cycles as normal management functions (Figure 6.8).

The idea of meta-management can be illustrated by means of an example. The example 
that is taken, is the one of TCP [RFC 793: "Transmission Control Protocol", Postel 
J.B., 1981 ] and SNMPv2 (see Section 4.2).

The ellipse at the bottom of Figure 6.8 shows the primary functions, as defined by the 
TCP protocol. Management of these functions can be accomplished by reading and 
modifying certain TCP variables. The variables that are available to management are 
defined as part of a separate standard: the MIB-II. Managers may read and write MIB-II 
variables across a network. The rules describing how to transfer MIB-II values and the 
policy that may be used to assign to different managers different MIB-II access rights, 
are defined by the SNMPv2 standards (the ellipse in the middle of Figure 6.9).

Figure 6.9: Primary functions, management and meta-management

The operation of the SNMPv2 functions depend upon various SNMPv2 variables. 
Examples of such variables are:

* authentication keys
encryption keys
access control tables.

The values of these variables are determined and maintained by meta-management functions (ellipse at the top of Figure 6.9). For this purpose, these variables are organized as part of three distinct MIBs: SNMPv2 MIB, Party MIB, and Manager to Manager MIB. The values in these MIBs are used by the normal management functions; they are controlled by the meta-management functions. Some of these meta-management functions are also defined by the SNMPv2 standards.

6.6 Cyclic design versus life cycle

The previous sections discussed the cyclic addition of management functions and the growth of functionality with each successive cycle. After completion of several cycles, the realization at our disposal already meets most of the user requirements. Still, additional cycles will be needed to include the remaining functionality. As compared to the first cycles, completion of these additional cycles may be time consuming for the following reasons:

- The designer may not yet have sufficient experience to solve all remaining issues. This is not surprising if we realize that traditionally most research is invested in basic functions (such as data transfer), and not (yet) in derived functions (such as management).

- The number of details that must be addressed in the later cycles is, as compared to the earlier cycles, usually much higher.

Addressing all tedious little details takes a lot of time. From a financial and competitive point of view it is therefore understandable that a wish exists to use the 'incomplete' realizations that have been developed thus far. The designer should anticipate this and allow quick and dirty solutions for these tedious little details; the perfect solutions should be postponed until later cycles.

A possible strategy is to let human operators perform the remaining management functions after the start of the operational phase (explicit management). In the specific case of the network layer (Figure 6.6) this means for example that some of the routing functions will be performed by persons. Since human beings can not be at multiple places at the same time, management should be performed in a centralized way.

6.6.1 From explicit to implicit management

The idea of human operators performing management functions was already discussed in Subsection 1.3.1. In that subsection it was shown that explicit management has several deficiencies, such as being expensive, causing a relatively high number of errors and
showing poor response times. Explicit management should therefore not be seen as the ultimate solution. After the network becomes operational the designer should therefore continue his job and improve the design!

An obvious step is to gradually automate the functions performed by the human manager. This implies that there will be a shift from explicit management to implicit management. We may say that the manager functions that were previously realized in 'brainware', will step by step be realized in hardware and software.

This process can be explained in terms of the cyclic design model. Figure 6.10 shows, for instance, the additional design cycle that is performed during the operational phase to improve the manager system. Although the figure shows only a single cycle, it can be expected that in reality multiple cycles will be needed. Note that the additional cycle does not comprise a new architectural and implementation phase; only the realization of the manager system will be improved.

![Figure 6.10: Realization of a better manager system](image)

This is because the changes should be restricted to the manager system; the managed systems should not be affected. The protocol should therefore remain the same; there is no need for an additional architectural and implementation phase.

It is important that already during the design phase there is some awareness that this shift from explicit to implicit management is likely to happen. To enable future modifications the designer may for instance decide to develop the manager system in the form of a management platform that can easily be extended with new management applications.
Management platform

Many of the manager systems that are on the market today are actually management platforms (Figure 6.11). To a certain extent these platforms can be compared to MS-DOS PCs: they can both perform elementary functions without additional software but they both become more powerful after special application software has been added. Such application software may be developed by independent vendors, who have particular expertise in certain problem areas.

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Management Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP</td>
<td>Openview</td>
</tr>
<tr>
<td>IBM</td>
<td>Netview 6000</td>
</tr>
<tr>
<td>SUN</td>
<td>SunNet Manager</td>
</tr>
</tbody>
</table>

Figure 6.11: Some existing management platforms

A typical management platform consist of the following components:

- A user interface (based on X11 or MS-Windows) to enter management commands and display management information.
- Communication software (such as SNMP or CMIP).
- A database system to log management information.
- Programming interfaces to add management software at a later stage. The programming interfaces should be 'open', which means that software may be written by different vendors.
- A 'script' facility to combine multiple management commands.

It may be expected that future management platforms will also include some kind of 'expert system'. Such systems allow a gradual shift from explicit to implicit management; several papers have already been written on this subject [SEE REFERENCES].

6.6.2 From centralized to distributed management

Centralized management is initially needed to ensure that human operators will be able to perform their job from a single location (or just a small number of locations). However, as management functions are automated step by step, the need to perform management in a centralized way slowly disappears. In Subsection 1.3.2 it was
explained that centralized management has several deficiencies (e.g. single points of failure, potential performance bottleneck). To solve these deficiencies, it is likely that at certain moments in time (after the shift from explicit to implicit management) the decision is made to distribute some of the management functions. There is thus also a shift from centralized to distributed management too (Figure 6.12).

![Figure 6.12: From explicit & centralized to implicit & distributed management](image)

A good example to demonstrate the shift from centralized to distributed management is the development of routing protocols for data communication networks. Initially the contents of forwarding tables was determined at a central place. Presently this function is implemented in a distributed way and the contents of the forwarding tables can be determined by the various routers within the network. A number of routing protocols have been standardized to define the management interactions between these routers.

The cyclic design model can also be used to illustrate this development (Figure 6.12). At the right side of the figure the design cycle is shown in which (parts of the) management functionality is distributed over the various systems. Such distribution requires a modification of existing protocols, which means that (as opposed to the redesign of the manager) an additional implementation and presumably also architectural phase will be needed.

This kind of redesign affects many network systems. Since not all systems may have been written down, the frequency at which these cycles are initiated may not be to high.

6.7 Conclusions

The cyclic design model can be used to illustrate the design of (service, protocol and element) management functions. According to this model, primary functions will be developed in the first design cycles. As part of the definition of primary functions, management functions will be identified. To keep the complexity of each design cycle within reasonable bounds, the elaboration of management functions will be postponed until later cycles.
The complexity of management functions may be such, that also the management functions should be managed (Section 6.5). To denote this 'management of management', the term meta-management is introduced.

The designer may not always be in the position to get complete grip of all management functions before the start of the operational phase (Section 6.6). In such case the designer should take a number of decisions to allow a human operator to perform the remaining management functions during the operational phase. One of these decisions is to select the explicit and centralized management approach. After the designer has obtained additional experience, this decision may be adjusted and step by step the approach may be changed into an implicit and distributed way.

The cyclic design model can also illustrate the various redesigns that may be needed because of this change in the operational phase. Whereas each cycle in the design phase results into a new prototype, each cycle in the operational phase results into a new system generation.