A Formal model for Total Quality Management

Saskia C. van der Made-Potuijt
Erasmus University Rotterdam
vandermade@few.eur.nl

H. Boudewijn Bertsch
Erasmus University Rotterdam
bertsch@few.eur.nl

Luuk P.J. Groenewegen
Leiden University
luuk@wi.leidenuniv.nl

December 20, 1996

Abstract

Total Quality Management (TQM) is a systematic approach to managing a company. TQM is systematic in the sense that it is uses facts through observation, analysis and measurable goals. There are theoretical descriptions of this management concept, but there is no formal model of it. A formal model can give a very precise description of the concept and will be usefull in organisations that consider to use TQM. Furthermore it can give organisations that have adopted TQM already more insight into their own situation.

We will use the formal modelling method Paradigm as this method can give a clear description of systems in which communication plays an important role. To be more precise, Paradigm is a modelling method for parallel behaviour that shows in detail the interaction among components.

We present two models in this paper: one for reactive improvement, both with and without a role for a seperate manager, and a model for proactive improvement. The models are not normative in the sense that companies should exactly use TQM as we describe it in order to do it successfully. At the other hand the models are also not theoretical in the sense that they give a precise description of the theory concerning TQM as this is known of books and articles. Moreover, by making the Paradigm models we had to add details to TQM that are not described in informal descriptions of books and articles.

The application of Paradigm shows much more detail in the management of the improvement processes. It provides insight as to when a manager using TQM can and probably should make choices and decisions regarding the improvement process. Furthermore, the models show that precise communication between the parties involved is of crucial importance to apply TQM.

- Catagories and Subject Descriptors:
  D.2.10[**Design**]-*representation*
  D.4[**Coding and Information Theory**]-*formal models of communication*

1 Introduction

The aim of this paper is to model Total Quality Management (TQM) with help of the Paradigm approach and to assess what we can learn both from the content of the model and the process of the modelling exercise. TQM is a systematic way of managing a company based on a number of working principles. Because of its systematic approach to work and improvement, TQM can be modelled using a formal method.

We use the Paradigm approach since this modelling method can clearly describe the different roles of a system and can model relationships and communication among components of a system. A Paradigm model consists of a number of sequential components that are represented graphically as a directed graph. Furthermore the interaction of these components is modelled by distinguishing subprocesses and describing managementprocesses. These aspects of the Paradigm method are essential and make it appropriate for modelling TQM since this approach emhasizes communication and relationships. A extensive explanation of Paradigm can be found in [Groe]. Paradigm is an intergated part of the software process modelling technique Socca, that is still under development at the University of Leiden [GELG93].

The paper is structured as follows. We will continu this introduction with a description of the principles of TQM. An example is provided of an invoice process to illustrate how TQM works in practice. The introduction will be concluded with a description of the construction process of a Paradigm model. After this, we will then use the Paradigm method for reactive improvement (with and and without a manager) and for proactive improvement.

For both situations we will discuss what we have learned from the modelling exercise. We will end with the conclusion and indicate future research.

1.1 The TQM principles

TQM is based on the following principles which aim to influence individual and group behaviour in alignment with agreed company objectives.

1. Management by fact through observation, analysis, experimentation and procedures using the Standard-Do-Check-Act (SDCA) workcycle for daily management and Plan-Do-Check-Act (PDCA) workcycle for improvement

2. Permanent improvement by reactive and pro-active improvement of both processes and product/services of the organization.

3. Management by process and result. This assumes that workprocesses are analysed, improved and used to move the organization and its members towards agreed objectives. Processes are analysed, improved and used using management by fact (principle 1)

4. Design, implementation, maintenance and improvement of systems which greatly influence behaviour such as remuneration, promotion, budget-allocation, communication and training.

5. Optimization of stakeholder value through application of the aforementioned principles notably:

   Customers (Satisfaction of articulated and unarticulated needs)

   Suppliers (Involvement and improvement)

Employees (Involvement and satisfaction)

Shareholders (Permanent Value enhancement)

Applying all these principles throughout a company leads to the label of a company *Managed by TQM*. In our attempt to model TQM we will predominantly restrict ourselves to the core principles, principle 1 and 2. In order to show how the application of the principles work in practice we provide an example. A more extensive explanation of TQM can be found in the [SGW]

1.2 Example

Assume there is an invoice process for which a workgroup is responsible and that the process of input-throughput-output, i.e. information for the invoice, actually processing this information and the output, the actual invoice to be send to the customer itself, is a 20-step process. In other words there is a SDCA-cycle of 20 steps. Each step in the proces is an opportunity for defect. Assume also that in each invoice there are 10 opportunities for defects i.e. the date, name and address of the customer, purchase order numbers, price, etc. Assume further that there are on average 25 defects per 100 invoices and that as a result of this customers delay payment and the company therefore loses valuable cashflow.

Reactive improvement with a manager In this situation, reactive improvement means that as a result of dissatisfaction with the defects, the workgroup asks approval from the manager to start the improvement cycle of this process. Improvement in this situation can also be characterized as defect-driven improvement. After approval from the manager, the workgroup starts the PDCA improvement cycle by analyzing the process using flowcharts, prioritization of defects and simulates alternatives. Assume that the workgroup creates an improved process, which is the main cause of defects, which consists of 10 steps thereby reducing the opportunity for defects in the input-throughput-output process. Subsequent analysis shows that the average defects per 100 invoices drops to 15. After approval of the manager the new standard operating procedure is implemented as a new SDCA-cycle.

The notion of permanent improvement or continuous improvement becomes apparent if the workgroup after some time of working with the new SDCA- cycle again asks for approval to start a PDCA-cycle with the aim of further reducing the number of defects. This could for example lead to making invoices on a computer using a database of customer address information thereby further eliminating defect opportunities. By systematically and continuously running the SDCA- and PDCA-cycles the number of defects can be eliminated to a zero-defect process.

Reactive improvement without a manager Reactive improvement without a manager merely eliminates the need to ask for approval to start a PDCA-cycle and sanction a new SDCA-cycle. However, in practice this means that employees must be empowered to use resources such as time and money to manage their SDCA- and PDCA-cycles. Typically this means that they must also have a higher level of managerial capabilities. Often these capabilities are acquired through training and experience and a less formal decision making process in the organization within certain boundaries.

Proactive improvement Proactive improvement can refer to a situation whereby top management initiates a goal-setting process through negotiating with lower hierarchical levels of the organization improvement projects. This is part of what is called a policy deployment process. Through this process improvement projects are aligned with the strategy of the company. The process is not initiated by dissatifaction or defects but primarily by an aspiration to achieve company objectives.

In this case a clear distinction is made between so called breakthrough projects, which are typically multi-functional and involve different departments (workgroups) and departmental or smaller incremental projects. For example, it could also mean that a zero-defect paper-based invoice process is improved as a result of a top management request into a fully electronic-based process. The end result could be that a number of departments with their particular SDCA-cycles would cease to exist.

The above example serves to illustrate the three situations in which we apply the Paradigm method to TQM.

1.3 The construction of a Paradigm model

A Paradigm model is constructed in a stepwise manner.

- A Paradigm model consists of a number of interacting processes. Each of the processes are represented graphically as a directed graph. The processes together form the parallel process.

  In this first model we develop there are three interacting processes, the SDCA- and PDCA-cycles and a Manager that is involved in the coordination of the two cycles.

- In order to be able to model the communication among these constituent components, the processes are split into parts, the so called *subprocesses*. This splitting has to be done in way that a subprocess is a temporary behaviour restriction of a process, valid between the receiving of two subsequent communications.

  For reactive improvement we will distinguish subprocesses for the SDCA- and the PDCA-cycle.

- A trap of a subprocess is restriction of the subprocess, marking its readiness to receive communication. By entering a trap, a subprocess explicitily asks for the next subprocess. A trap cannot be left as long as this subprocess is prescribed.

  In the subprocesses for the SDCA- and the PDCA-cycle we will also indicate the traps.

- The control of communication can be modelled by a seperate decision process, a so called *manager process*. This process coordinates the behaviour of the various objects. The objects of which the behaviour is controlled by a manager process are called the employee processes of that manager process.

  In our model the *manager process* will be the Manager that we have modelled. The SDCA- and PDCA-cycles are the employee processes.

- The manager process can formally be expressed means of a so called *state-action interpretor*. A state-action interpreter is a function defined on the states and actions of the manager process and the values are the subprocesses and traps. To be more precise, the function values of the states of the manager are the subprocesses for its employee processes and the function values for the transitions are the traps of the subprocesses. This function can be expressed graphically by a so called *labelling function* that labels each state and transition of the manager process with the values of the state-action interpreter.

The state-action interpreter for reactive improvement will be the description of the Manager extended with the subprocesses that it prescribes for its employee processes in its states and the traps of the subprocesses of the SDCA and PDCA in its transitions.

2 A Paradigm model for reactive improvement

In this section we present the first Paradigm model. We start with a description of Quality Management by Reactive Improvement that anticipates already on the Paradigm model. Then we will develop the Paradigm model for the situation where a separate Manager is involved in the coordination of the SDCA- and PDCA-cycles. After this we will present the Paradigm model for reactive improvement without a separate manager.

2.1 Quality management by reactive improvement

The SDCA and PDCA-cyclus together are called the Deming-cycles and can be represented as follows:

The SDCA-cycle The PDCA-cycle

Figure 1: The Deming-cycles.

The interpretation of the stages of the cycle of process control for daily management, the SDCA-cycle is as follows:

S Have a standard

D Do the standard

C Check the results

A Act to return to the standard or to initiate reactive improvement

So there are two types of action SDCA-cycle can take in A: to continue working according to the standard and to initiate reactive improvement. Furthermore we assume there are two more interactions of the SDCA-cycle with its environment: the SDCA-cycle has to be started and a new standard can be adopted. The PDCA-cyclus for reactive improvement has the following stages:

P Plan a solution, this involves selecting a theme, collecting and analyzing data etc..

D Do, implementing the solution to test it.

C Check, the PDCA is evaluating the results of the tests.

A Act to standardize the solution.

There are two interactions with the environment: the PDCA-cycle will have to be started and the PDCA-cycle will present a new standard for the SDCA- cycle. In the description until now is valid for situations were a manager is involved in and for situations were no manager is necessary.

In this first model we devellop, we assume that there is a Manager involved in the coordination of the two cycles. The main tasks of this Manager are:

1. To start the SDCA-cycle, this will be done by sending a *Start SDCA* message to the SDCA-cycle.

2. To receive a request for reactive improvement from the SDCA-cycle. This we will call the *Request RI* message. This message will contain an exact description of the problem, including the exact figures that were the reason why a request for reactive improvement was sent.

3. to start the PDCA-cycle, this will be done by sending a *Start PDCA* message to the PDCA-cycle

4. The PDCA-cycle will present a new standard for the SDCA-cycle to the Manager, this will be done with a *Pres NS* message, the abbreviation of Presenting the New Standard. This message is not only an indication of the fact that a solution to the problem has been found, but it also contains a precise description of the New Standard.

5. To send the message *Impl NS* to the SDCA-cycle in order to implement the new standard.

6. After adopting the new standard, the SDCA-cycle will send the message *done* to the Manager.

So the actions of the Manager involve sending messages to the SDCA- or PDCA-cycles and reacting on received messages.

In the next paragraph we will give the description of the three components involved: a SDCA-cycle, its corresponding PDCA-cycle and a Manager. The behaviour of these components is described in directed graph where the messages among the components are sent in the transitions of the diagram. The states represent rather passive behaviour like preparing to send messages etc.. The states itself are also represented as a transition, as there is some action in a state, but this action in a state does not involve communication with other components.

2.2 The model for a SDCA-cycle

The SDCA-cycle is not active before the Manager sends a *Start SDCA* message. This non-activity is reflected by state 0. The states 1, 2, 3 and 4 represent the SDCA stages of the cycle as described in the previous section. The transitions 1→2, 2→3 and 3→4 do not involve interaction with another component. If the cycle requests reactive improvement it expresses this wish by sending the message *Request RI* to the Manager in transition 4→5.

In state 5 it waits for the New Standard and it receives the command *Impl NS* in transition 5→6. In state 6 the new standard is implemented. If the cycle does not request a reactive improvement, it continues with the standard. In the diagram this is reflected by transition 4→1.

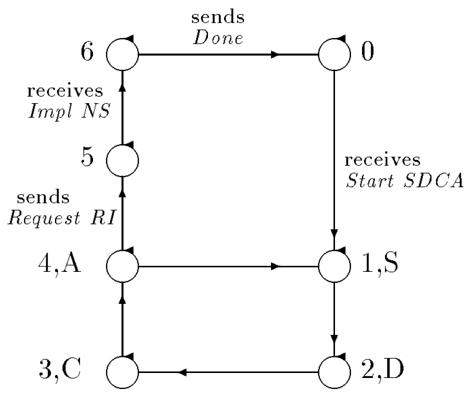

Figure 2: The behaviour of a SDCA-cycle.

0 The SDCA-cycle is not active yet.

0→1 The SDCA-cycle receives a *Start SDCA* message from the Manager and is activated.

1 The SDCA-cycle has a standard.

1→2 The SDCS-cycle is going to use the standard.

2 The SDCS-cycle is working according to the standard.

2→3 The SDCA-cycle is ready with the standard

3 The SDCA-cycle is checking the results of the execution of the standard.

3→4 The SDCA-cycle is finished checking the results.

4 The SDCA-cycle is deciding wich action has to be taken: to continu because the results were in compliance with the set marges or to initiate reactive improvement.

4→1 The SDCA-cycle has decided it can continue with the execution of the standard.

4→5 The SDCA-cycle sends the message *Request RI* to the Manager.

5 The SDCA-cycle has notified the Manager that reactive improvement has to be initiated and waits for the result from the Manager.

5→6 The SDCA-cycle receives the message *Impl NS* from the Manager.

6 The SDCA-cycle is implementing the new stand- ard.

6→0 The SDCA-cycle sends the message *Done* to the Manager.

We include the messages that the SDCA-cycle receives and sends as labels at the edges.

If in practice reactive improvement is requested in an organization, the SDCA-cycle will not wait for a reaction of the PDCA-cycle but continues with the daily work of the SDCA-cycle. This can also be modelled in Paradigm but we have not done this for the following two reasons: this is the first Paradigm model for many of the readers, therefore we did not want to make it too complicated and secondly, later we will present the Paradigm model for proactive management in which there is this parallel behaviour of the daily work and the improvement cycle.

2.3 The model for the corresponding PDCA-cycle

Before the Manager sends a *Start PDCA* message, the PDCA-cycle is not active. This will be reflected by state 0. The states 1, 2, 3 and 4 represent the PDCA stages for reactive improvement as described in the paragraph 2.1. The PDCA-cycle presents the new standard to the Manager by sending the message *Pres NS* in the transition 4→0.

0 The PDCA-cycle is not active yet.

0→1 The PDCA-cycle receives the message *Start PDCA* from the Manager.

1 The PDCA-cycle is planning a solution, this involves selecting a theme, collecting and analyzing data, analyze the causes.

1→2 The PDCA-cycle is going to implement the solution in order to test it.

2 The PDCA-cycle is implementing the solution to test it.

2→3 The PDCA-cycle is ready with the testing of the solution.

3 The PDCA-cycle is evaluating the results of the tests and preparing a new standard.

3→4 The PDCA-cycle is finished checking the results.

4 The PDCA-cycle has defined a new standard for the SDCA-cycle.

4→0 The PDCA-cycle sends the *Pres NS* message to the Manager.

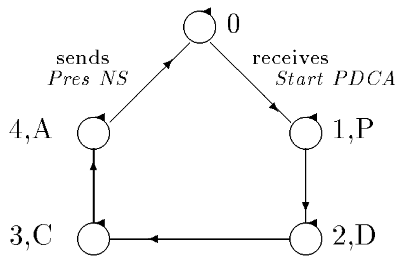

Figure 3: The behaviour of a PDCA-cycle.

In practical situations there is more iteration, for example there will be a transition 3→2 to be able to test another version of the new standard.

2.4 The model for the Manager

Also for the Manager we introduce a state 0 in which the Manager is not actively involved in the management of this particular combination of SDCA- and PDCA-cycle. Its activity starts with the sending of the message *Start SDCA*.

0 The Manager is not active yet.

0→1 The Manager sends the message *Start SDCA*.

1 The Manager is waiting for a reaction of the SDCA-cycle.

1→2 The Manager receives the message *Request RI*

2 The Manager is preparing to activate the PDCA-cycle.

2→3 The Manager sends the message *Start PDCA* to the PDCA-cycle.

3 The Manager is waiting for the results of the PDCA-cycle.

3→4 The Manager receives the message *Pres NS* from the PDCA-cycle.

4 The Manager prepares to implement the new standard.

4→5 The Manager sends the message *Impl NS* to the SDCA-cycle.

5 The Manager is waiting for the *Done* message from the SDCA-cycle.

5→0 The Manager receives the message *Done* from the SDCA-cycle.

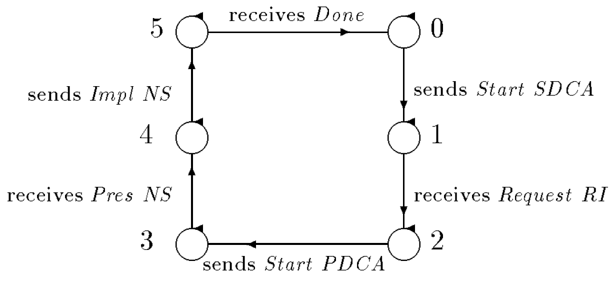

Figure 4: The behaviour of a Manager.

By including the messages as labels of the transitions the figure shows very clear that we modelled the Manager in a way that it enters other states by sending and receiving messages.

2.5 The communication in reactive improvement with a manager

In Paradigm the communication between the components of a system is modelled by means of subprocesses, traps and a manager process. In our model we will give the Manager the role as manager process controlling the behaviour of the SDCA- and PDCA-cycles. To this aim we will structure the subprocesses for the SDCA- and PDCA-cycles in a way that these cycles can only change the current subprocess into a new one, after receiving a message from the Manager. This paragraph is organized as follows. In paragraph 2.5.1 we will distinguish the subprocesses for the SDCA-cycle and in 2.5.2 we will do this for the PDCA-cycle. The communication between the Manager and the two cycles will be made precise in 2.5.3 by giving the state-action interpreter.

2.5.1 The subprocesses of the SDCA-cycle

We distinguish two subprocesses for the SDCA-cycle that are represented by their corresponding graphs in figure 5.

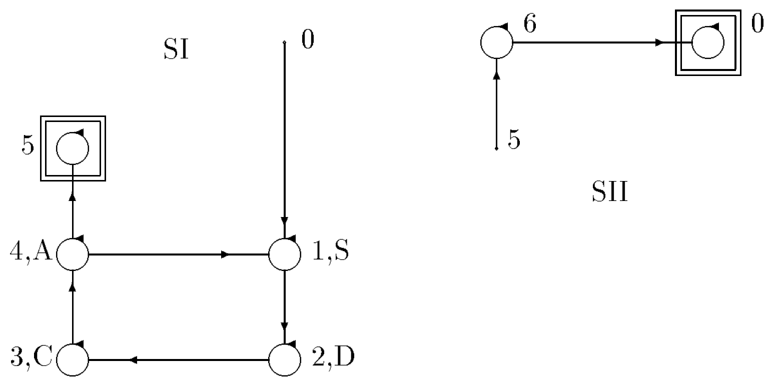

Figure 5: The subprocesses of a SDCA-cycle.

In SI the SDCA-cycle has been activated and can continu until it enters the trap. The trap, {5} indicates the state in which the SDCA-Cycle can receive the message *Impl NS*. Subprocess SII is described after the SDCA-cycle has received this message. The trap {0} of this subprocess indicates the SDCA-cycle is prepared to receive the message *Start SDCA*. Subprocess SI will be entered after receiving this message.

2.5.2 The subprocesses of the PDCA-cycle

We also distinguish two subprocesses for the PDCA-cycle that are represented by their corresponding graphs in figure 6.

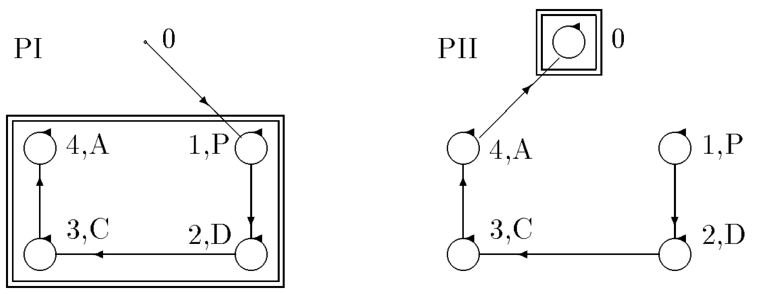

Figure 6: The subprocesses of a PDCA-cycle.

The Manager activates the PDCA-cycle by describing subprocess PI. After this it will immediately prescribe subprocess PII, as only in this subprocess the PDCA-cycle can send the message *Pres NS*.

2.5.3 The state-action interpreter for reactive improvement with a manager

To complete the Paradigm model we will now discuss the control of the communication. As said before we give the Manager the role of coordinator of its subordinate processes, the SDCA- and the PDCA-cycle. How they exactly depend on each other can formally be expressed by a *state- action interpreter*. A state-action interpreter is a function defined on the states and actions of the manager process and its values are the subprocesses and traps. In our example in the states of the Manager, the Manager prescribes the subprocesses for the SDCA- and PDCA-cycles and in the transitions the traps of the subprocesses of the cycles are indicated. The interpretation of a transition labelled by a trap is as follows: the transition can only be made after the subprocesses are in the trap.

The state-action for our model is expressed graphically by the labelling function reflected in figure 7.

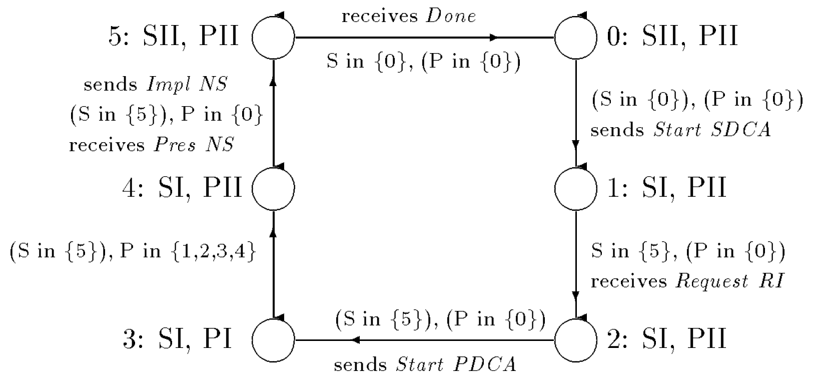

Figure 7: The Manager as state-action interpreter.

This function can be interpreted as follows. In state 0 the Manager has not given a command *Start SDCA* yet, so SII and PII are prescribed in which these cycles are not active. The transition 0→1 can only be taken if the subordinate process SDCA is in trap {0} of subprocess SII. Subordinate process PDCA is then in trap {0} of subprocess PII. It is there already for some time and the fact that it is there is not relevant for the transition, therefore we write this trap in brackets.

In state 1 the Manager prescribes subprocess SI for the SDCA-cycle, indicating it has been activated. The PDCA-cycle has not been activated yet in state 1, so subprocess PII is still prescribed. If the SDCA-cycle enters trap {5} in subprocess SII, transition 1→2 is possible for the Manager. The PDCA-cycle is still in trap {0} of subprocess PII.

In state 2 the Manager still prescribes subprocess PII for the PDCA-cycle as the message *Start PDCA* has not been sent yet. For transition 2→3 no traps of the subprocesses have to be entered In state 3 subprocess PI is prescribed for the PDCA-cycle. The Manager can make transition 3→4 if the PDCA-cycle is in trap {1, 2, 3, 4}. No message as we described them in paragraph 2.1 are sent in this transition, only a message to the PDCA-cycle that prescribes subprocess PII. In state 4 the Manager prescribes subprocess PII in which the Manager can receive the *Pres NS* message. Notice that this is a difference with the description of the Manager in figure 4. There we assumed the Manager receives the

Pres NS message in transition 3→4, but for we state-action interpreter this transition is neccessary to notice that the PDCA is in trap { 1,2,3,4} of subprocess PII. The message *Pres NS* is received by the Manager if the PDCA-cycle is in trap {0} of subprocess PII and the Manager can immediately send the message *Impl NS* to the SDCA-cycle. So two messages are sent in transition 4→5. We can adapt the model for the Manager by adding one extra state between the states 4 and 5, to make it possible that only one message is sent in one transition. For the management of the cycles this adaption is not necessary. Therefore we do not add a state, only the description of the transitions.

In state 5 subprocess SII is prescribed for the SDCA-cycle in which the new standard is adopted. The transition 5→0 is possible for the Manager if the SDCA- cycle has entered trap {5} of subprocess SI, the PDCA-cycle will still be in trap {0} of subprocess.

2.6 A Paradigm model for reactive improvement without a manager

In this paragraph we will give a variation of the Paradigm model for reactive improvement in which we will assume that the Deming-cycles will communicate directly instead of communicating via a Manager. As the communication is now directly between the SDCA- and PDCA-cycles there are less messages necessary, only the messages *Request RI* and *Pres NS*.

The description of the SDCA-cycle that we developped for the reactive improvement with a manager needs to be slightly adapted and we will call this new cycle a SDCA′-cycle. This SDCA′-cycle will not receive a *Start SDCA* message anymore from a manager, but we still assume that in state 0 the cycle is not active. We only give a description of the adapted states and transitions.

0→1 The SDCA′-cycle is activated.

4→5 The SDCA′-cycle sends the message *Request RI* to the PDCA′-cycle.

5 The SDCA′-cycle is waiting for a reaction from the PDCA′-cycle.

5→6 The SDCA′-cycle receives the message *Pres NS* from the PDCA′-cycle.

6→0 The SDCA′-cycle has adopted the New Standard.

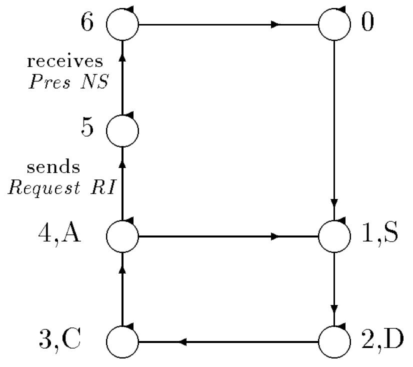

Figure 8: The behaviour of a SDCA′-cycle.

For the PDCA-cycle the only difference is that the message *Request RI* is received directly from the SDCA-cycle and the message *Pres NS* is directly sent to the SDCA-cycle.

We will therefor not give a graph for its behaviour. We will call the new PDCA-cycles PDCA$'$.

In the communication between the SDCA$'$- and PDCA$'$-cycle we will give the SDCA$'$ the role of the manager. To this aim we will structure the subprocesses for the PDCA$'$-cycles in a way that these cycles can only change the current subprocess into a new one, after receiving a message from the SDCA$'$-cycle. As the message that is sent by the SDCA$'$-cycle to the PDCA$'$-cycle, *Request RI* is equal to the message sent by the Manager in the Paradigm model for reactive improvement with a Manager, we can distinguish exactly the same subprocesses for the PDCA$'$-cycle as we did for the PDCA-cycle in figure 6. The communication between two cycles is made precise by describing the state-action interpreter. As the manager role is done by the SDCA$'$- cycle we define a state-action interpreter for the SDCA$'$-cycle.

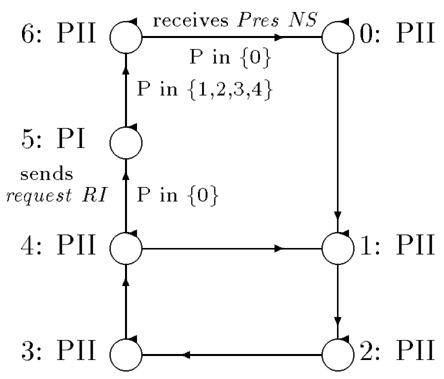

Figure 9: The SDCA$'$-cycle as state-action interpreter for the PDCA$'$- cycle.

This function can be interpreted as follows. In state 0 the PDCA$'$-cycle has not been activated by the SDCA$'$-cycle and therefore the SDCA$'$-cycle prescribes PII for the PDCA$'$-cycle. This is also valid for the states 1, 2, 3 and 4. In the transition 4$\rightarrow$5 the SDCA$'$-cycle sends the message *Request RI*. This can only be received by the PDCA$'$-cycle if this cycle is in trap {0} of subprocess PII. In state 5 the SDCA$'$-cycle prescribes subprocess PI, indicating the PDCA$'$ cycle has been activated. The message *Pres NS* can only be received by the SDCA$'$-cycle, if the PDCA$'$-cycle is in subprocess PII. Therefore subprocess PII is prescribed as soon as the PDCA$'$-cycle has entered trap {1,2,3,4}. So in state 6 the SDCA$'$ prescribes PII again.

Notice the difference with the description of the behaviour of the SDCA$'$- cycle in figure 8. The message *Pres NS* that the SDCA$'$ receives in transition 6$\rightarrow$0 was received by the SDCA-cycle in transition 5$\rightarrow$6. The reason for this later receiving of the message is, that the SDCA$'$ uses one transition for describing another subprocess of its employee process.

The model for the SDCA$'$ cycle can be adapted by adding a state 7 in which the New Standard can be implemented. For the management of the cycles this adaption is not necessary and without complications we can assume the new standard is adopted in state 0. Therefore we do not add a state, only the description of the transitions.

In the model of reactive improvement with a manager, the situation was rather similar, were the state-action interpreter used one transition to subscribe another subprocess for one of its employee processes. Also there an extension with an extra state was not necessary for a good management of its employee processes.

2.7 Discussion reactive improvement

By making the Paradigm model one has to make clear what parties are involved and how they behave. Allthough the wellknown Deming-cycles show a part of the behaviour of the SDCA- and PDCA-cycle, they do not describe the complete behaviour including the messages that are to be sent among the parties involved. We have described these messages, and have given a more complete description of the Deming-cycles that integrates the sending and receiving of these messages.

The coordination of the components can only be done by sending these messages. Therefore a good structure of these messages and a good protocol for the order in which they are sent is essential for a successfull TQM management for reactive improvement.

Furthermore we see that a Paradigm model for reactive improvement without a Manager has many similarities with the model that includes a manager. There are less messages necessary, and compared to the model with the manager, the two cycles have more responsibilities. This increased responsibility involves more complex decisions in the cycles.

In both situations we saw that the manager tasks does not only consist of sending the messages, but also of influencing directly the behaviour of the subordinate processes. This is the reason that both manager processes even needed to be adapted a little.

The models show more precise than is usually done in books and articles, the structure of the communication and the minimal contents of the messages that is neccessary for reactive improvement. At the other hand it does not give to much details that better can be filled in for the specific situation in an organization.

As reactive improvement is a less complicated form than proactive improvement we hope that the description of proactive improvement will help to make TQM more concrete than pure theoretic descriptions on that rather complicated form of improvement.

3 A Paradigm model for proactive improvement

In this section we present a Paradigm model for proactive management. First we will give a description of the process of proactive improvement, then we will develop the Paradigm for this process.

3.1 Quality management by proactive improvement

As discussed in the introduction, proactive improvement is initiated by top management and departmental or workgroup behaviour emerges as a result of a negotiating process with top management. The top management will usually not be involved themselves in the details of the proactive management process, but a team will be installed to perform most of the tasks. This team, that we will call the PI-Team, an abbreviation of Proactive Improvement Team will translate the objectives to plans for all the involved SDCA-cycles and communicate with them. The top management, that we will call in our model Top Management (TM), will not influence the behaviour of the SDCA-cycles directly, only through the PI-Team. We assume that if reactive improvement is requested by a SDCA-cycle this is done without a manager. Therfore, we do not have to take into consideration the interaction of the PI-Team or the Top Management with the PDCA-cycles.

So there are three types of behaviour that we will describe in the Paradigm model. The behaviour of the Top Management, the PI-Team and a SDCA-cycle. There are usually many SDCA-cycles active in the process of proactive improvement, but only one Top Management and one PI-Team.

The Top Management initiates the goal-setting process and installs a PI-Team. It sends its objectives to the PI-Team and expects a reaction from the PI-Team in the form of a so called *Concept Plan* for proactive improvement. In this way the Top Management negotiates with the PI-Team about the exact goals and how to reach them. After receiving the *Concept Plan*, the Top Management prepares a *Definite Plan* that the PM-Team has to implement in the organization. In this *Definite Plan* are only little marges that the PI-Team can use in the implementation of the plan.

The PI-Team receives the objectives from the Top Management and develops concept plans for SDCA-cycles in order to achieve the goals. After collecting reactions from the SDCA-cycles and combining the reactions, the PI-Team sends the *Concept Plan* to the Top Management. It receives the *Definite Plan* from the Top Management and translates this into definite plans for the SDCA-cycles. In order to be able to set the marges right it collects a reaction from the SDCA-cycles on the definite plan and then, without consulting the Top Management, gives the command to start working according to the new situation.

The SDCA-cycles remain responsible for the daily work during the process of improvement. Furthermore they must respond to requests of the PI-Team concerning a reaction on the concept plan and the definite plan.

We start with a more detailed description of the behaviour of the Top Management, and proceed with the behaviour of a PI-Team that is responsible for the plans and implementation of the proactive improvement. Then we will describe the consequences for the SDCA- and PDCA-cycles. We will conclude with the interaction of the involved components.

3.2 The model for the Top Management

The Top Management of a company is always active, but not always in relation to proactive management. We will give a description of its behaviour that is related to proactive management. As explained, we will model the situation in which the TM will only communicate with a PI-Team, so there is no direct communication with the SDCA or PDCA-cycles. The description starts at state 0 where the TM is not actively involved in proactive management.

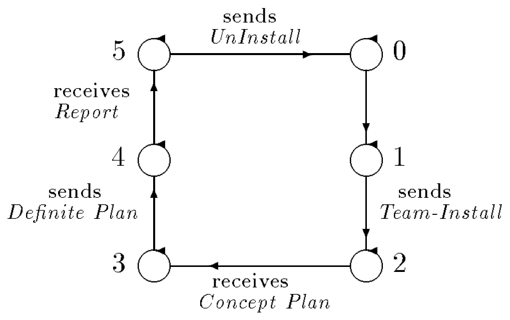

Figure 10: The behaviour of the TM.

0 The TM is not actively involved in proactive management.

0→1 The TM is activated to initiate proactive management.

1 The TM is preparing objectives and a PI-Team that will manage the proactive improvement.

1→2 The TM installs a PI-Team by sending the message *Team-Install*. The objectives for the proactive improvement are included in this message.

2 The TM is waiting for a reaction from the PI-Team in the form of a concept plan.

2→3 The TM receives a *Concept Plan* for proactive improvement from the PI-Team.

3 The TM is evaluating this Concept Plan and prepares its reaction on it.

3→4 The TM reacts to the PI-Team by sending a *Definite Plan* for proactive improvement. The PI-Team has to install this plan in the organization and has only little marges to adapt it.

4 The TM is waiting for the PI-Team to install the plan.

4→5 The TM receives a *Report* from the PI-Team concerning the installation of the definite plan.

5 The TM is reflecting on the report it received.

5→0 The TM stops reflecting on the report and sends the message *UnInstall* to the PI-Team.

3.3 The model for the PI-Team.

Before the Top Management sends the message *Team-Install*, the PI-Team is not activated. After it has received this message, including the objectives for proactive management, the PI-Team makes a plan of how to reach the objectives set by the TM. It determines the consequences of this plan for all the existing SDCA's.

It is possible that some SDCA's will certainly not be affected by the proactive improvement process. Therefore we will make a distinction between two classes of SDCA'S. A SDCA is of

Class A if it certainly will not be affected by the proactive improvement

Class B if it might be affected by the proactive improvement.

Later it may become clear that there are SDCA's of *Class B* that are also not affected, but in this early stage of proactive improvement this might not be certain yet. One might also be even more carefull at the beginning of proactive improvement and assume that there are no SDCA's in *Class A*.

The PI-Team makes concept plans for each of the SDCA's of *Class B*. We will call the plans sent to all these SDCA-cycle(i) *Concept Plan(i)*. These SDCA-cycles of *Class B* are after receiving this concept plan not allowed anymore to request for reactive improvement. The reason for this is twofold: an improvement of the existing process might not be used anyhow and the effort must be used for collecting data for the PI-Team and reacting on the *Concept Plan(i)* and later on the *Definite Plan(i)*.

We assume that the SDCA-cycles of *Class A* will receive a message *CAC*, an abbreviation of *Class A Continu*, indicating they are not affected by the proactive improvement process. The SDCA-cycles of *Class A* will continu during the whole process of proactive improvement and can ask for reactive improvement by sending a *Request RI* to their PDCA-cycle. In practice the management of a company can choose to forbid to request for sometime to the SDCA's of *Class A* in order to be able to use more effort for the process of proactive improvement.

The SDCA-cycles of *Class B* have to give a reaction to the PI-Team on the *Concept Plan(i)*. This is done in the message *Reaction(i)*. After collecting all the reactions from the SDCA-cycles and reflecting on these reactions, the PI-Team sends a *Concept Plan* to the TM. The TM will send a *Definite Plan* back to the PI-Team that has to be implemented by the team. There are only small marges in this *Definite Plan* that can be filled in by the PI-Team. To this aim the PI-Team will collect reactions from the involved SDCA-cycles on the *Definite Plan*.

The PI-Team will translate this *Definite Plan* into definite plans for each of the involved SDCA's. These will be called *Def-Plan(i)*. We assume that there has been made no mistake in the classification *Class A* and *Class B*, so the definite plans will only have to be send to SDCA's of *Class B*. In *Class B* three types of SDCA's can be distinguished:

Type 1 are the SDCA's that will not be affected by the process of proactive improvement,

Type 2 are the SDCA's that will be adapted and

Type 3 are the SDCA's that will cease to exist.

The new standards for the SDCA-cycles of *Type 2* and *Type 3* will not be developped by a PDCA-cycle as described for reactive management, but by special *test*-cycles. In the next paragraph we will give a complete description of these test-cycles. For some period, that will be given precisely in the below description, the SDCA's of *Type 2* and of *Type 3* will be active in parallel with their test-cycles.

Characteristic for the SDCA's of *Type 3* is, that in the new situation the function of several SDCA's of this type will be done by one newly developped SDCA. This new SDCA will replace several old SDCA's of *Type 3*. We assume that all but one of the constituent SDCA's will be made inactive and that one SDCA of *Type 3* will remain active and adopt the new standard. It is possible that in one process of proactive improvement several clusters of SDCA-cycles of *Type 3* will be replaced by one new SDCA-cycle, one new SDCA-cycle for every cluster.

To the SDCA's of *Type 1*, the message *T1C*, an abbreviation of *Type 1 Continu* will be send at the moment that to the SDCA-cycles of *Type 2* and *Type 3* the *Def-Plan(i)* will be send. This implies for the SDCA-cycles of *Type 1* that they can continu as before, including that they can ask for reactive improvement.

In the *Def-Plan(i)* a reaction from the SDCA's is asked by the PI- Team in order to be able to set the marges of the definite plan right. The message of *Def-Plan(i)* will be send to

1. the existing SDCA's of *Type 2*
2. the test versions of the SDCA's of *Type 2*,
3. the existing SDCA's of *Type 3* and
4. the test versions of the SDCa's of *Type 3*

These 4 kinds of SDCA's will be called *involved*.

After having sent the message *Def-Plan(i)* the two versions of the SDCA's of *Type 2* will be active in parallel, the existing version and the test version. The SDCA's of *Type 3* remain active and the test-cycles of them are active as well. The SDCA's of *Type 1* receive the message that they can continu, including the possibility for reactive improvement.

The reactions of the involved SDCA's on the definite plans are called *ReDef(i)*. After collecting the reactions on the definite plans, the PI- team sends the message *Continu* to the involved SDCA's. Then it decides on the last details of the definite plan and sends the message *Start New Situation* to the involved SDCA's. This is a command indicating that the SDCA's of *Type 2* from now on have to adopt the new standard developped by the test version. The test version is made inactive and reactive improvement can start for these SDCA-cycles of *Type 2*.

Some of the SDCA's of *Type 3* that receive the command *Start New Situation* will be made inactive and the others will adopt the standard developped by the test-SDCA's. The test-SDCA's wil be made inactive as well. Reactive improvement is from now on possible for the SDCA's of *Type 3*.

After having sent a *Report* to the TM, the PI-Team can be uninstalled by the TM.

The behaviour of the PI-Team is reflected in figure 11.

0 The PI-Team does not exists, but the TM is preparing one.

0→1 The PI-Team receives the message *Team-Install* including the objectives for proactive improvement from the TM.

1 The PI-Team is preparing a concept plan that will be send to the SDCA-cycles of *Class B*.

1→2 The PI-Team sends the messages *Concept Plan(i)* to all the SDCA-cycles of *Class B* and the message *CAC* to those of *Class A*.

2 The PI-Team is waiting for the respons of the SDCA-cycles to these concept plans.

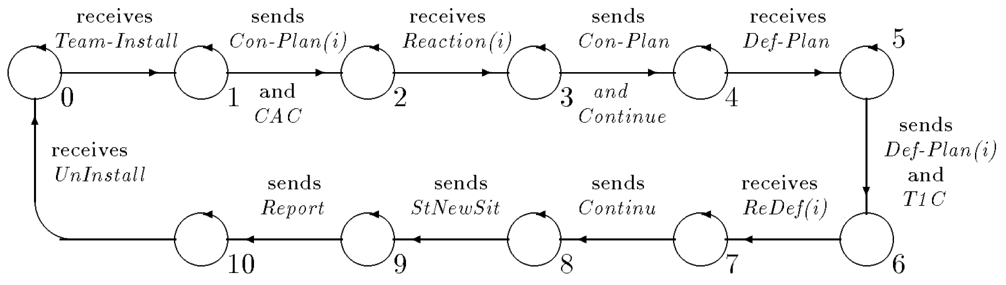

Figure 11: The behaviour of the PI-Team.

2→3 The PI-Team has received all the messages *Reaction(i)*.

3 The PI-Team is evaluating the reactions and preparing a Concept Plan for the Top Management.

3→4 The PI-Team sends the *Concept Plan* to the TM and a *Continu* message to the SDCA's of *Class B*. The SDCA's of *Class B* continu without the possiblity for requesting reactive improvement.

4 The PI-Team is waiting for the reaction of the TM to the plan.

4→5 The PI-Team receives the *Definite Plan* from the TM.

5 The PI-Team prepares the definite plans for the SDCA's.

5→6 The PI-Team sends the definite plans *Def Plan(i)* to the involved SDCA's.

6 The PI-Team is waiting for the *ReDef(i)* messages from the involved SDCA-cycles.

6→7 The PI-Team has received all the reactions on the definite plans.

7 The PI-Team is going to send the message *Continu* to the involved SDCA-cycles in order to let the work continue.

7→8 The PI-Team sends *Continu* message to the involved SDCA-cycles whithout the possiblity for requesting reactive improvement.

8 The PI-Team is reflecting on the reactions of the SDCA-cycles to the definite plan and preparing to make the new situation final.

8→9 The PI-Team sends the message *Start New Situation* to the involved SDCA's.

9 The PI-Team is preparing to send a respons to the TM.

9→10 The PI-Team sends the *Report* to the TM.

10 The PI-Team is waiting for a reaction from the TM.

10→0 The PI-Team receives the message *UnInstall*.

Notice that in transition 3→4 the PI-Team sends a message to the SDCA's and to the TM. This could also have been modelled differently, by means of two transitions. This would not have changed the model in an essential way. In the transitions 8→9 and 9→10, we have used two transitions in a similar situation which shows how little difference this makes.

There are two moments that the PI-Team sends the message *Continu* to the SDCA-cycles, in transition 3→4 and in transition 7→8. We can give these messages the same name as the effect is equal, continu without the possibility for reactive improvement.

3.4 The models for the SDCA- and SDCAWR-cycle

The SDCA-cycle has to be adapted to make it possible for these cycles to send a reaction to the PI-Team. We use the SDCA's developped in the model for reactive improvement without a manager as the behaviour to be adapted, as this was the more mature behaviour. The adapted SDCA-cycle will simply be called SDCA-cycle. An individual SDCA-cycle will be called SDCA-cycle(i).

The adaption must make it possible for SDCA-cycle(i) to send a reaction to the PI-Team on the *Concept-Plan(i)* and on the *Def-Plan(i)*. This will be done by one extra state that will be called $4'$. Furthermore one transition, 4→0 is added that reflects the situation in which a SDCA-cycle is made inactive. The behaviour of this SDCA-cycle is reflected in the left part of figure 12.

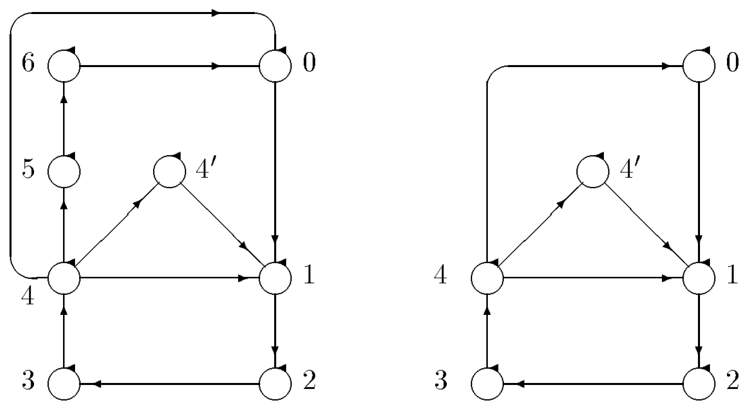

Figure 12: The behaviour of the SDCA- and SDCAWR-cycle

We will give the description of the new state and the new transitions of the SDCA-cycle.

4→4′ The SDCA-cycle has received the message *Concept Plan(i)* or *Def-Plan(i)* and has collected data in order to give a reaction. In this transition the message *Reaction(i)* or *ReDef(i)* is sent to the PI-Team.

4′ The SDCA-cycle has sent its reaction and is waiting for a reaction from the PI-Team to continu again.

4′→1 The SDCA-cycle receives the message *Continu* from the PI-Team.

4→0 The SDCA-cycle is made inactive due to the message *Start New Situation* from the PI-Team.

As explained in the previous paragraph, we assume that for each SDCA-cycle there is another cycle, similar to an SDCA-cycle, that can perform the tests in order to develop a new standard for the SDCA-cycle. These *test* SDCA-cycles will have a restricted behaviour compared to the SDCA-cycles as they will not be able to ask for reactive improvement. We will call these *test* SDCA-cycles SDCAWR-cycle, an abbreviation of SDCA-Without-Reactive improvement. There are not only SDCAWR-cycles for each of the SDCA-cycles, also for the testing of new SDCA-cycles that will replace several SDCA's of *Type 3*. The SDCAWR-cycles are always present, but not always active.

For the SDCAWR it is also necessary to be able to give a reaction to the PI-Team, therefore state $4'$ and its related transitions are present, but not the state 5 and 6 in which it requests reactive improvement and applies the new standard.

We assume that the SDCA-cycles initiate reactive improvement without the interference of a manager. Therefore the behaviour of the PDCA-cycles and the interaction with the SDCA- cycles will not be different than we described it in the related section, section 2.6. A consequence of this is, that the PI-Team will never have to communicate with a PDCA-cycle direcly, this will always be done by the SDCA-cycle in a way we described in section 2.6. In a indirect way the PI-Team does influence the PDCA-cycles as the PI-Team indicates whether it is possible for a SDCA-cycle to request reactive improvement or not.

3.5 The model for the communication in proactive improvement

In the management of proactive improvement there are two levels of communication. At the highest level the Top Management and the PI-Team communicate, at the lower level the PI-Team and the SDCA- cycles communicate. For the communication at the highest level we will give the Top Management the manager role. For the exact modelling of this interaction between TM and the PI-Team we will distinguish subprocesses for the PI-Team in section 3.5.1. In section 3.5.2 we will give the description of the communication between the TM and the PI-Team.

At the lower level, the PI-Team will be manager for the SDCA- cycles. The communication of the PI-Team with the PDCA-cycles is indirect, therefore we will only distinguish subprocesses for the SDCA-cycles. We do this in section 3.5.3. and in 3.5.4. we give the description of the communication between the PI-Team and the SDCA- and PDCA-cycles.

3.5.1 The subprocesses of the PI-Team.

We distinguish three subprocesses for the PI-Team related to the TM, that are represented by their corresponding graphs in figure 13.

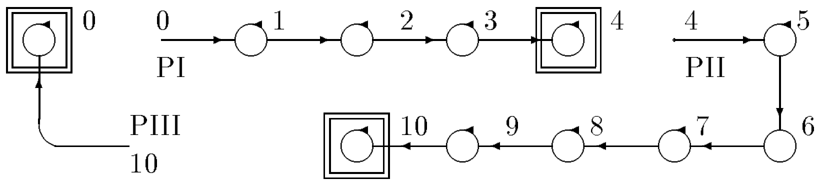

Figure 13: The subprocesses of a PI-Team.

In PI the PI-Team has been activated by the TM and has sent the *Concept Plan* to the TM. In the trap of this subprocess, {4} it waits for the *Definite Plan* from the TM. In PII it has received the *Definite Plan* from the TM and gives the appropriate messages to the SDCA's involved to install the new situation. In the trap, {10}, of this subprocess it has sent a *Report* to the TM and waits for a reaction of the TM. As we see, the three subprocesses are constructed in a way that they can only be left after a message from the TM.

3.5.2 The communication between the TM and the PI-Team

We will now discuss the control of the communication between the TM and the PI-Team. The communication is expressed by the state-action interpreter represented in figure 14.

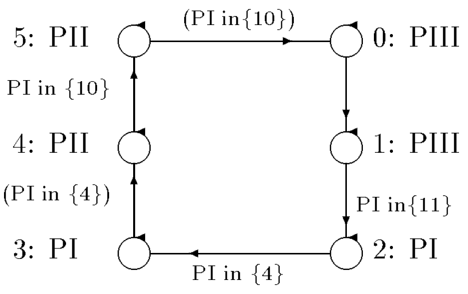

Figure 14: The TM as state-action interpreter for the PI-Team.

The interpretation of this state-action interpreter is as follows:

In state 0 and 1 no PI-Team is installed, therefore PIII is prescribed. State 2 can only be entered if the PI-Team is in trap {11} of subprocess PIII, which means an old PI-Team has been uninstalled. In the states 2 and 3 the TM prescribes subprocess PI indicating the PI-Team is installed, this team has made a plan, has communicated with the SDCA's and sent a *Concept Plan* to the TM.

The TM can only enter state 3 if the PI-Team is in trap {4} where the PI-Team sends a *Concept Plan.* In state 4 and 5 the TM prescribes subprocess PII, indicating the definite plan has to be implemented and the *Report* has to be sent to the TM.

State 5 can only be entered if the PI-Team is in trap {10} of subprocess PII where the PI-Team has sent a *Report.* The TM can only send a *UnInstall* message in transition 5→0 to the PI- Team if the PI-Team is still in trap {10} of subprocess PII.

We have put brackets around the traps that were entered before the transition and therefore are not relevant to this particular transition.

3.5.3 The subprocesses of the SDCA- and SDCAWR-cycles

We distinguish four subprocesses of the SDCA-cycles that are represented in figure 15. The first subprocess, S0, represents the behaviour of a SDCA-cycle for reactive improvement as we described in the previous section. It reflects the behaviour of a SDCA-cycle that did not receive a *Concept Plan(i)* yet. After receiving this message subprocess SI is entered.

SII will be entered after the SDCA-cycle has received the *Continu* message from the PI-Team. The SDCA's of *Type 1* will enter SO after they receive in subprocess SII the message *Type 1 Continu.* The SDCA's of *Type 2* and *Type 3* cycle receive the messages

Def-Plan(i) in SII. Also the new versions of the SDCA's of *Type 2* and the *New* SDCA's are activated by this message.

After the SDCA's of *Type 2* and *Type 3* have received the message *Def-Plan(i)* in SII, they enter SI in which they will send the message *ReDef(i)* to the PI-Team. After these SDCA's have received the message *Continu* they enter SII. The SDCA's of *Type 2* in subprocess SII that receive the message *Start New Situation*, will adopt the new standard that has been test by the corresponding SDCAWR.

The SDCA's of *Type 3* that are in subprocess SII will receive the message *Start New Situation*. Some of them will adopt the new standard developped by the test SDCA's and continu. These SDCA's will enter subprocess SO. Other SDCA's of *Type 3* that receive this message will be made inactive and subprocess SIII is entered.

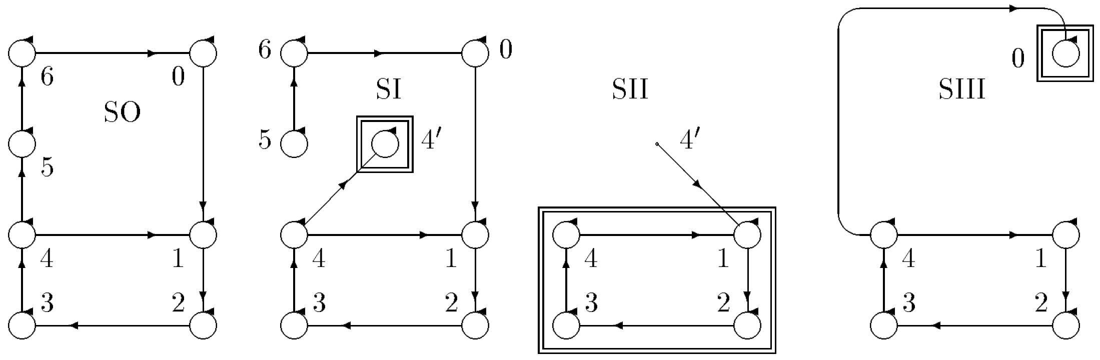

Figure 15: The subprocesses of the SDCA-cycle.

We distinguish three subprocesses of the SDCAWR that are represented in figure 16. In the first subprocess, S0′, the SDCAWR- cycle is not active as it did not receive a *Def Plan(i)* yet. After receiving this message subprocess SI′ is entered. Remember that only SDCAWR's of *Type 2* and *Type 3* exist. The PI-Team expects a *ReDef(i)* from these SDCAWR-cycle's. SII′ will be entered after the message *ReDef(i)* is sent to the PI- Team and the SDCAWR-cycle has received the *Continu* message.

If in subprocess SII′ the SDCA's receive the message *Start New Situation*, they enter subprocess SO′ and become inactive.

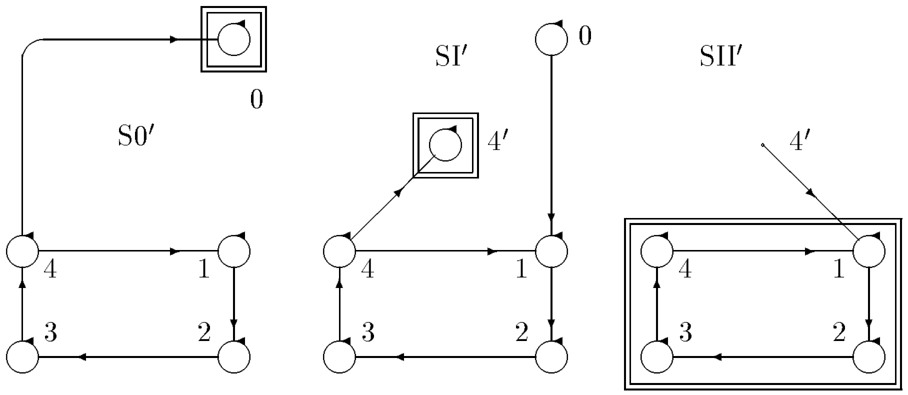

Figure 16: The subprocesses of the SDCAWR-cycle.

3.5.4 The communication between the PI-Team and the SDCA- cycles

The control of communication between the PI-Team and the SDCA- cycles will be expressed as usual by a state-action interpreter. In this case the state action interpreter is a rather complex function because for the differents classes and types of SDCA-cycles the values of the function can be different. Furthermore the subprocesses for the SDCAWR- cycles have to be given as well. We will therefore not use a labelling function to describe the state-action interpreter, but a table. In the left column of the table all the states and transitions of the PI-Team are given. The other columns represent the different classes and types of SDCA-cycles. The SDCA-cycles of *Class B* are divided into the substituant components, type 1, 2 and 3. In each state the PI-team prescribes subprocesses for all types of SDCA-cycles and these are given in the tabel in the corresponding column. The subprocesses for the corresponding SDCAWR-cycle given after the '/'.

	Class A	Type 1	Type 2	Type 3
0	SO/SO′	SO/SO′	SO/SO′	SO/SO′
0→1				
1	SO/SO′	SO/SO′	SO/SO′	SO/SO′
1→2				
2	SO/SO′	SI/SO′	SI/SO′	SI/SO′
2→3		SI in {4′}	SI in {4′}	SI in {4′}
3	SO/SO′	SI/SO′	SI/SO′	SI/SO′
3→4				
4	SO/SO′	SII/SO′	SII/SO′	SII/SO′
4→5				
5	SO/SO′	SII/SO′	SII/SO′	SII/SO′
5→6		SII in {1,2,3,4}	SII in {1,2,3,4}	SII in {1,2,3,4}
6	SO/SO′	SO/SO′	SI/SI′	SI/SI′
6→7			S and S′ in {4′}	S and S′ in {4′}
7	SO/SO′	SO/SO′	SI/SI′	SI/SI′
7→8				
8	SO/SO′	SO/SO′	SII/SII′	SII/SII′
8→9			S and S′ in {1,2,3,4}	S and S′ in {1,2,3,4}
9	SO/SO′	SO/SO′	SO/SO′	SO-SIII/SO′
9→10				
10	SO/SO′	SO/SO′	SO/SO′	SO-SIII/SO′
10→11				
11	SO/SO′	SO/SO′	SO/SO′	SO-SIII/SO′

The interpretation of this function is as follows. In state 0 and 1 SO is prescribed for all SDCA-cycles and they continu as in the case of reactive management. Their corresponding test-cycles are not active, subprocess SO′ is prescribed for them. For the SDCA's of *Class A* these subprocesses remain prescribed during the whole process of proactive improvement.

In the first part of the function, before state 6, subprocess SI is prescribed for the SDCA's of *Class B* and subprocess SII. This indicates they have to give a reaction to the *Concept Plan*. After SDCA's of *Class B* have entered trap {4′} of subprocess SI, the PI-team can prescribe SII in state 4. Their corresponding test SDCA-cycles are not activated.

In transition 5→6 the PI-Team sends the message *Type 1 Continu* to the SDCA-cycles of *Type 1* and *Def-Plan(i)* to the involved SDCA's. This can only be done if the SDCA-cycles of *Class B* are in trap { 1, 2, 3, 4} of subprocess SII.

In the second part of the function, from state 6 onward, the SDCA-cycles of *Type 1* can continu as before, therfore subprocess SO is prescribed for the them and they can request again for reactive improvement. They are not influenced anymore by the process of proactive improvement.

In this second part of the function, the test versions of the SDCA-cycles of *Type 2* and *Type 3* are activated. In state 6 and 7 the two versions of the SDCA-cycle of *Type 2* are active, for the existing SDCA's subprocess SII is prescribed, for the test versions subprocess SI'.

State 8 can only be entered by the PI-Team if the involved SDCA's have entered trap $\{4'\}$ of their subprocesses, indicating they have given a reaction on the *Definite Plan(i)*. In transition 8→9 the PI-Team sends the *Start New Situation* message to the involved SDCA's. This implies that the SDCA's of *Type 2* will adopt the new standard and that subproces SO is prescribed for them in state 9. The corresponding test-SDCA's are made inactive by prescribing SO' for them. This is also valid in the states 10 and 11 of the PI-Team.

The *Start New Situation* message that is sent in transition 8→9 to the SDCA's of *Type 3* implies that some of the SDCA's will adopt the new standard and that subproces SO is prescribed for them in state 9. Other SDCA's of *Type 3* will be made inactive by prescribing SIII for them. The corresponding test- cycles for both sorts of SDCA's of type 3 are made inactive by prescribing SO' for them. This is also valid in the states 10 and 11 of the PI-Team

3.6 Discussion proactive management

The modelling exercise for proactive management clearly demonstrates the complexity involved in sending and receiving communication messages. In other words it shows how difficult it is to manage company-wide improvement projects.

Again we had to make clear what parties are involved and how they behave. In the literature there are not such complete descriptions of the messages that have to be sent between the TM, the PI-Team and the SDCA-cycles. We have described these messages, and have given a description of the components involved that integrates the sending and receiving of these messages.

Furthermore we have seen that it is necessary to make a distinction between the two classes of SDCA-cycles and a further disctinction into the different types. The reason for this is, that SDCA-cycles will behave differently in the process of proactive improvement.

The model shows that for the development of new SDCA's there are test- cycles necessary. The PDCA's that correspond to a SDCA-cycle are in general not always fit to develop a new standard that can accomplish the function of several existing SDCA's.

The Paradigm model provides insight wat which points the management can make choices as to allow reactive improvement or not for the SDCA-cycles in the organization, being involved in proactive improvement or not.

Still there are remain many details that can be filled in for the specific situation in an organization that applies proactive improvement.

4 Conclusion and future research

The modelling exercise forces precise definitions of relationships and actions of the TQM approach and thus helps to clarify this management philosophy. At the other hand it does not describe details that better can be filled in appropriate for a specific situation in an organization. The formal model can therefore be very helpfull for organizations that are considering to use TQM and for organizations that use already TQM it can give more insight into their situation.

As this formal approach of TQM, based on the Paradigm formalism has given so much more insight into the processes involved in TQM, we want to use another formal method for a different view on TQM.

Recent research in (software) process modelling has resulted in combining Paradigm with object-orientation. This has led to the process modelling language SOCCA [GELG93], [GELG94]. We will present a SOCCA model for TQM in a forthcoming paper. Such a SOCCA model can specify parallel behaviour within one component, e.g. the SDCA controller, more easily. Moreover, SOCCA provides a higher level of standardization in the choices of manager processes, employee process, subprocesses and traps. This turns out to be very useful in setting up a model.

Contents

List of Figures

References

[BeWi95] H.B.Bertsch, A.R.T. Williams, *Kwaliteitsmanagement als besturingsconcept.* In: TAC, Tijdschrift voor Tijdschrift voor Accountancy en Controlling, pp 46-49, juni 1995.

[BeWi96] H.B.Bertsch, A.R.T. Williams, *Hoe topmanagers een transformatieproces besturen.* In: M&O Quarterly, pp 3-27, nr.1 1996.

[Groe] L.P.J.Groenewegen, *Parallel Phenomena, a series consisting of technical reports. 1986-1990* Department of Computer Science, University of Leiden.

[GELG93] G.Engels, L.P.J.Groenewegen, *SOCCA: Specifications of Coordinated and Cooperative Activities.* University of Leiden, Department of Computer Science, 1993.

[GELG94] G.Engels, L.P.J.Groenewegen, *SOCCA: Specifications of Coordinated and Cooperative Activities.* In: A.Finkelstein, J.Kramer and B.Nuseibah (eds.), Software Process Modelling and Technology, John Wiley & Sons Inc., pp 71-102, 1994.

[KH92] K.Hosotani, Japanese Quality Concepts, Quality resources, White Planes, 1992.

[SGW] S. Shiba, A. Graham, D. Walden, *A New American TQM* Center for Quality Management, Portland, Oregon, 1993.

æ