Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR-TP-2002-602

A model for Procedural Knowledge

ir. Niels Basjes

UNIVERSITEIT NYENRODE
THE NETHERLANDS BUSINESS SCHOOL

Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR-TP-2002-602

A model for Procedural Knowledge

ir. Niels Basjes

This paper was written as the thesis required for completing
the *Parttime Doctoraal Programma*, a part-time Masters degree
in corporate business, at Nyenrode University in Breukelen.

Customer:	National Aerospace Laboratory NLR
Owner:	National Aerospace Laboratory NLR
Division:	Information and Communication Technology
Distribution:	Unlimited
Classification title:	Unclassified
	December 2002

Preface

This paper was written as the thesis required for completing the *Parttime Doctoraal Programma*, a part-time Masters degree in corporate business, at Nyenrode University in Breukelen. I would like to thank everyone who has supported me the last three years during what has been a great learning experience. I would like to especially thank my parents for looking after me, Wout Loeve for his support in getting started, Ton Hazelaar and Geert Jan Dekker for understanding working and studying is not an easy combination and last but not least Sunil Choenni, Robin Bakker and Aad van Dorp for their support in writing this thesis.

I thank you all.

Niels Basjes

Executive Summary

Knowledge Management has become an important issue in current businesses [Ref. 1]. It's purpose is to retain and reuse the available knowledge within the business. One way to preserve knowledge is to encode experiences and knowledge in procedures within the organisation.
A procedure is simply the description of a path from 'Situation A' to 'Situation B'. Several advantages can be mentioned for this policy.

- Procedures contain the solutions for problems that have been solved before. Therefore a person performing a task using a procedure, won't have to 'reinvent the wheel' anymore. From the perspective of the company this means more work can be done in the same time with less problems.
- New employees are capable to perform a required task more quickly and with less training if they are guided by a procedure.
- If more tasks can be performed in the same time, the total costs of documenting, training personnel and performing the tasks can be reduced.

In this paper a model is designed that allows procedural knowledge to be stored and retrieved in an automated system.

Many procedures in businesses can be represented in a procedural or in a declarative form.
A procedural representation is useful when a procedure is relatively static and the level of complexity is limited. A declarative representation is useful in a more dynamic situation when fragments of procedures need to be reordered depending on the problem that needs to be solved. Many procedures can be modelled very well in either a procedural or declarative way.

In the three examined cases, one fictitious and two found at the National Aerospace Laboratory, some procedures were observed that would benefit from a more hybrid approach where procedural and declarative representations can be mixed together. The existing tools for managing procedures that have been examined use either the procedural or the declarative representation, but not both.

Based on these observations a model has been designed that allows mixing both procedural and declarative representations for representing a procedure. This model allows for a more flexible representation of business procedures. This additional flexibility results in a wider range of possible applications than the examined existing tools have shown.

To what extent this model is more practically applicable is a point for further research.

Contents

(47 pages in total)

1 Introduction

Knowledge Management has become an important issue in current businesses [Ref. 1]. It's purpose is to retain and reuse the available knowledge within the business. One way to preserve knowledge is to encode experiences and knowledge in procedures within the organisation.
A procedure is simply the description of a path from 'Situation A' to 'Situation B'. Several advantages can be mentioned for this policy.

- Procedures contain the solutions for problems that have been solved before. Therefore a person performing a task using a procedure, won't have to 'reinvent the wheel' anymore. From the perspective of the company this means more work can be done in the same time with less problems.
- New employees are capable to perform a required task more quickly and with less training if they are guided by a procedure.
- If more tasks can be performed in the same time, the total costs of documenting, training personnel and performing the tasks can be reduced.

At the National Aerospace Laboratory (NLR) many procedures have been documented and over time the size and complexity of these procedures has increased to the point where some procedures became too complex to handle. Within the IT department at the NLR the problem of information overload, a user is overwhelmed with too much information, became an important problem.

Therefore we set ourselves as goal to work towards a (practical) solution for this problem. To be more precisely, we focus on the improvement of the management and use of procedures, i.e., to the reduction of the information overload. This idea was born around the time I was looking for a thesis subject for my Masters degree at Nyenrode University. This paper is the result of that combined effort.

This paper will first describe the problem in more detail and the research question in Chapter 2. After what a procedure actually is and what it means to a company in Chapter 3, an overview of the related research will be given in Chapter 4. In Chapter 5 the cases will be described and their requirements for the model. The model itself is designed in Chapter 6. Chapter 7 will analyse if the model actually fits the needs from the cases. Finally in Chapter 8 the conclusions, future recommendations and ideas are described.

2 Problem description

2.1 Problem description

In current businesses a large number of tasks is described in terms of procedures [1]. Usually this is done to meet the requirements of ISO-9001:1994 or similar standards. The basic idea behind these ISO-9001:1994 certificates is

"Describe what you do and do what has been described".

These certificates imply that all repetitive tasks within an organisation have been described in such a way that anyone with sufficient prior knowledge can perform a specific task by following the provided procedure.

This practise has lead to two issues in practical situations:

1) **Information Overload**

   A very common practise of publishing procedures is as a complete document describing all knowledge regarding all uses of the procedure. Such a document is often published on paper or as a complete file on an intranet (usually in MS-Word, PDF or HTML format). An important consequence of publishing procedures as complete documents is the problem of information overload. Information overload means that knowledge that is not applicable to the task at hand is presented to the user. This situation is a very severe kind of information overload: all knowledge that is applicable to all the possible uses of the procedure is presented to everyone who needs to use any part of the procedure. Someone who only needs a small part of a procedure must go through the entire document making sure no relevant information is missed. This practise has proven to be very sensitive to errors; relevant parts of procedures are sometimes missed. In order to resolve this issue the need exists to filter the information to only what is needed for a specific task.

2) **Maintenance of copied knowledge**

   The procedure that must be followed in a specific situation is very often a mixture of procedures written by various parties, both internal and external. The issue that arises is that none of the authors of the procedures has full insight into the all required modifications in the existing procedures when a change is made. A procedure for a task is often constructed by *copying* parts of existing procedures. In practical situations this means that hardly ever all of the existing copies of a procedure are updated if the original procedure is updated. In order to resolve this issue the need exists to employ a Single Point of Definition (SPOD) strategy where existing knowledge is referred to instead of copied.

[1] Chapter 3 will explain in more detail what a procedure is.

2.2 Research question

The need has risen to make it possible to represent the knowledge contained in the procedures in such a way that the user of the knowledge is helped to find all the required knowledge for the task at hand. In this research this kind of knowledge is called “Procedural knowledge”. Not only should the user find all the knowledge required but the user should not find anything that is not relevant to the task at hand. This means that not only the precision[2] of the answer must be 100%, but also the recall[3] must be 100%.

The goal has been set to design an automated system that allows procedural knowledge to be stored and retrieved by anyone who needs it. In order to achieve this goal the first step is to design a model for procedural knowledge that allows this kind knowledge to be stored and retrieved in an automated system.

The research question for this thesis is to design this model in such a way that it is usable for most common procedures that appear in businesses.

[2] Precision: The percentage of the returned answer that is relevant to the task at hand.

[3] Recall: The percentage of the available knowledge to the task at hand that is actually returned.

3 Procedures and Procedural Knowledge

The primary question that needs to be answered before we can analyse the given cases is what do we actually mean with a procedure and procedural knowledge?

3.1 Purpose

The purpose of a procedure is to guide a person through the maze of available choices to get to the required end situation in minimal time. A procedure contains the solutions to problems that have been solved before. The ever-changing environment ensures that situations will change and thus maintaining procedures to correct for these changes is essential. Problems in either the procedure itself or because of the changing environment are likely to occur. As soon a problem is detected and solved, the applicable procedure is updated to reflect the new 'fix'. The next time the procedure is used the person executing the procedure won't have to 'reinvent the wheel'. In the short run this means spending extra time documenting this fix but in the long run much time will be saved. This is however only true if everyone uses the available procedures [4]. From the perspective of the company this means more work can be done in the same time with less problems. Also new employees are capable to perform a required task more quickly and with less training if they are guided by a procedure. If more tasks can be performed in the same time, the total costs of documenting, training personnel and performing the tasks can be reduced.

3.2 Procedures and Sub-procedures

In general a procedure can be considered as a description of

"What to do to get from situation A to situation B?"

A procedure consists of one or more steps that have to be performed in the described order in order to reach a specified end situation. Each step of a procedure can be a procedure in itself; this is called a sub procedure. Note that this last definition can be applied recursively to a procedure and the sub-procedures it consists of.

A set of procedures can only be used if an external entity explicitly specifies the initial situation and a desired end situation.

This external entity can be a user, a computer, something else or a combination these.

[4] Solving this very serious motivation issue is left out of this thesis.

The terms “situation A” and “situation B” must be seen in broad perspective. Some possible examples of procedures:

- What to do to get from *“requirements”* to *“satisfied customer”*
- What to do to get from *“order”* to *“delivered goods and received money”*
- What to do to get from *“Amsterdam”* to *“Breukelen”*
- What to do to get from *“empty computer”* to *“productive workstation”*
- What to do to get from *“coffee bean”* to *“hot cup of espresso”* [5]

These examples show that the applications of procedures can be found in all parts of a business.

Note that if the knowledge does not exist, then it is not possible to write a procedure documenting this knowledge. For this reason the procedure on how to get from *“major losses”* to *“fortune 500 rating”* does not exist.

3.3 Decisions, rules and constraints

In almost all procedures the need to make decision exists; *“If the box weighs more than 30kg you are required to get a trolley to move the box”*. These kinds of decisions are derived from rules and regulations that have been drawn up by the people who knew the reasons why this rule was needed. In our example they knew that lifting heavy weights is bad for someone’s health.

The descriptions of when which procedure is to be performed and in what order with respect to the other procedures are called “rules” or “constraints”.

These rules define how the building blocks of the procedures can be glued together. This is an important part of the actual knowledge that is stored in the procedures.

3.4 Procedural knowledge

As mentioned in section 3.2, a procedure is a sequence of steps with decision points. The problem is that just storing the procedure in an automated system is in general a bad idea. Example: Assume we have a system that contains the procedure:

“When you enter the room you must turn on the light.”

[5] Very important in all kinds of business!

This seems like a plausible procedure because it contains information regarding the context *"When you enter the room"*. In the long run this procedure does not contain the knowledge regarding the reasons *why* you must turn on the light. The documentation only states that it is needed "*when you enter the room"*, but that wasn't the real reason. In this example the real reason was that the room didn't have any windows: the room is always very dark. If the reasons for writing a procedure change it is required to change or even remove the procedure. This is impossible without storing these underlying reasons for a procedure.

It is therefore important to store as much information regarding the scope a procedure is to be used in: When what it is within the scope it is used in. This information is the actual knowledge that defines what procedural knowledge really is.

4 Related research

4.1 Defining Procedural Knowledge

When searching for existing literature on “Procedural Knowledge” it became apparent that the term is sometimes used for something related but essentially different. Most Internet sites refer to “Procedural Knowledge” as a procedural representation of knowledge.
For example Deepak Kumar [Ref. 2] states

***Procedural Knowledge**: Knowledge is encoded in functions/procedures.*

Deepak Kumar further explains Procedural Knowledge as being the procedural representation of knowledge. Fred Nickols [Ref. 3] describes two definitions for Procedural Knowledge:

- *“knowledge that manifests itself in the doing of something”*
- *“knowledge about how to do something”*

Fred Nickols states that this second definition it is no different from a declarative representation. In this paper the term “Procedural Knowledge” is used as *“knowledge about how, when and why to do something”*. This definition allows for a better representation of the scope a procedure can be used in as described in section 3.4.

4.2 Generating technical documentation

N.Miliaev et al. [Ref. 4] describes a system with very similar requirements to what is needed for this thesis. The primary goal of their design is to automatically generate technical documentation in multiple languages. By explicitly defining each step as a procedure with an initial state and an end state the system can derive a procedure by recombining several steps into a complete procedure. Describing each step in such detail is needed because this knowledge is then used to generate the same documentation in multiple natural languages, in this case English and Russian. This system can be used to define many kinds of procedures but there is one major obstacle: Because the entire system is purely declarative, the kill level required for authoring a procedure is too high for common business procedures.

4.3 Procedural documentation

The Dutch company Mavim [Ref. 5] has created a software product that makes it easy to document business processes and make them available to the people that need to perform them. This product has been implemented in such a way that the procedures that are documented using this system are static. The procedures must all be explicitly defined manually. None of the procedures are generated in an automated fashion. This system provides the author a graphical representation of the procedure using a flowchart generated in Visio. Many business processes can be modelled in this way but not all. Note that this system only allows for procedures represented in a pure procedural form (see section 6.2 for more info on representations).

5 In search of the requirements

In this chapter we will try to find the requirements for the model that is to be designed in Chapter 6. First some business requirements are described in section 5.1. In the later sections the three available cases will be analysed. The first case (section 5.2) is fictitious. The other two cases (sections 5.3 and 5.4) have been provided by the NLR.

5.1 Business requirements

If the model for procedural knowledge is to be used successfully it must be applicable to at least all of the provided cases. If these cases are representative for all procedures in a business is doubtful, it is impossible to get an overview of all existing business situations in the time available for this thesis.

To solve the problem of information overload (mentioned in section 2.1) the option must exist to filter the knowledge depending on the task at hand.

In most businesses management wants to ensure all modifications to the procedures used to run the business. In the case of problems it is desirable to 'roll back' a change and return to the previous version. In order to achieve this version management and an approval workflow are needed.

Many of the people actually executing procedures in a business do not have a university degree in handling and describing procedures. The model must therefore allow for a user interface that can be understood by everyone.

To summarise these requirements:

1) The model must have enough expressive power to represent all the procedural knowledge that was found in the provided cases.
2) The model must make it possible to filter the knowledge based on the actual requirements depending on the task at hand.
3) The model must allow version management and approval workflow of the knowledge.
4) A person without university training should be able to learn how to use the model. This means that the way in which the knowledge is represented should be intuitive to any user writing or examining a stored procedure.

5.2 Replacing a light bulb

This is a fictitious case that has been created to start with a simple problem for illustrative purposes.

5.2.1 Description

When a person in a company is confronted with a light that is not working he/she will report this to the responsible person. The responsible person is confronted with the task that the light should be working again. For this task a procedure is needed to guide this person through all the required steps. Note that the procedure here presented has been oversimplified in order to illustrate a few of the situations that can occur. To visualise this simple procedure a flow chart was created that summarizes the procedure.

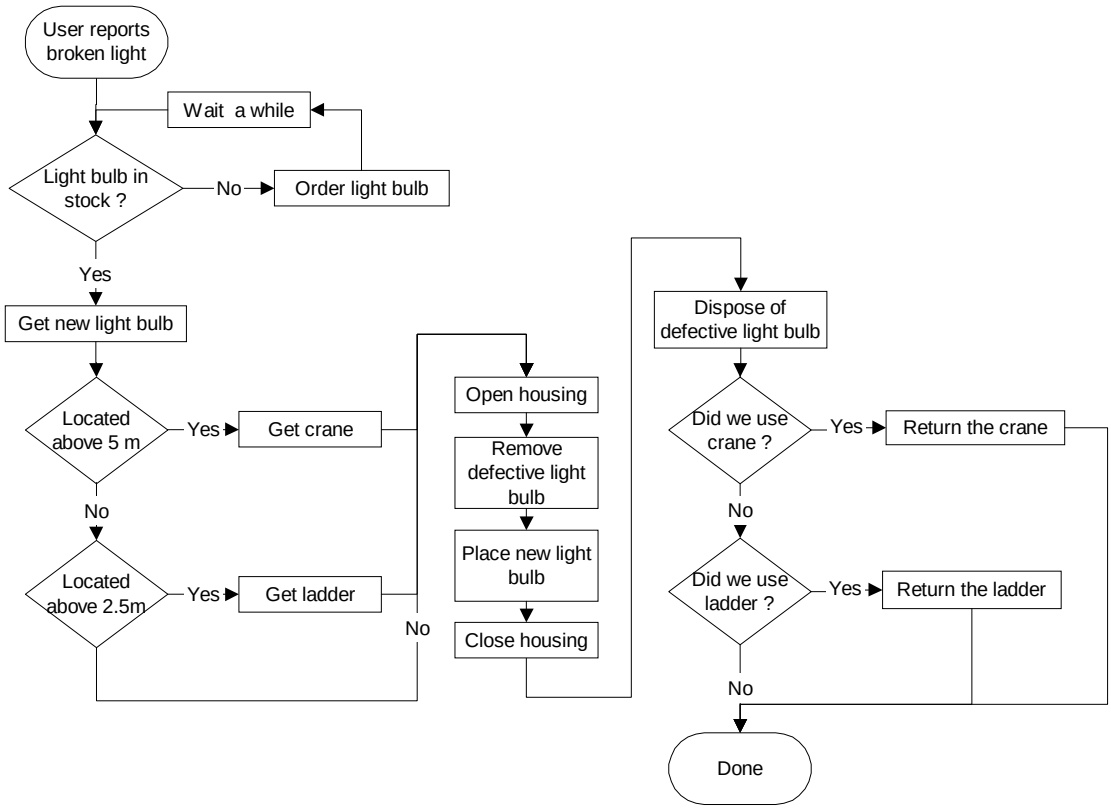

Figure 1 Workflow for replacing a light bulb

5.2.2 Types of knowledge and information

The procedure used by the mechanic depends on quite a few 'environmental' factors. A lot of questions need to be answered in order to determine which steps must be followed.

- Where are the light bulb and the light bulb housing located?
- Is the light bulb really broken or is something else wrong?
- What type of light bulb is needed?
- Is that specific type of light bulb in stock?
- How do I order a new light bulb?
- What is the cheapest point of sale for this type of light bulb?

The questions that need to be answered to fix a light bulb quickly overlap with other expertises, in this example purchasing.

The knowledge that directly applies to the replacing of a light bulb can be found in several sources. Among these sources are:

- The educational material used in schools for electrical mechanics.
- ARBO [6] rules which include the safety regulations regarding the person who is actually performing the task.
- NEN electrical norms [Ref. 6] define how an electrical installation must be implemented. These include rules like electrical wires should be insulated.
- The physical factors surrounding the workplace (height of the light bulb housing).

5.2.3 Types of procedures and relations

The first issue that becomes immediately clear is that some procedures can be reused in slightly different situations. The procedures *"Get a ladder"* and *"Get a crane"* can be reused for something other than replacing a light bulb. If for example a mechanic has to place a video camera somewhere then the same procedure must be available to reach the same goal: Getting to the work location safely.

The second issue that arises is that this case has sub-procedures that are related:

- "*get new light bulb"* and "*dispose of broken light bulb*"
- "*get a ladder"* and "*return the ladder"*
- "*get a crane"* and "*return the crane"*
- "*open housing*" and "*close housing"*

[6] ARBO: ARBeids Omstandigheden wet = The Dutch working environment regulations law.

These procedures are all in terms of *"If the step A is preformed then step B must be performed also before the user is done"*. They must always be executed in pairs or not at all.

Most steps require a specific situation before they can be executed

- To place the new light bulb the lamp housing needs to be empty.
- To place the new light bulb you must have a new light bulb.

5.2.4 Model requirements

The requirements for the model that come from this case are therefore:

1) It must be possible to reuse parts of a procedure in a different context.
2) It must be possible to model relationships between separate sub-procedures.

5.3 Software Installation Manuals

5.3.1 Description

At the NLR the decision was made to document the way software should be installed. This documentation must be so clear that anyone could follow the procedure and have a working installation afterwards. A Software Installation Manual is a document that describes the entire procedure for installing a single piece of software.

It is important to note that most Software Installation Manuals are related to the Microsoft Windows platform and related applications because this is the platform that is used the most at the NLR. The other common desktop platform at the NLR is Linux (>10% of the NLR desktop systems). For Linux only a few Software Installation Manuals have been written because most procedures for installing Linux have been documented directly in the form of installation scripts. These scripts combined with a management database make that extensive documentation on paper did not seem to have an added advantage.

5.3.1.1 A single application

The relevant part of a Software Installation Manual consist of the following chapters:

- **Licensing and support**
  This describes the type of licensing that applies to the specific application.
- **Assumptions**
  What are the assumptions that are required to install the software. In software engineering terms this would be referred to as the *pre-conditions*. Note that the term *post-condition* seems to be missing in the Software Installation Manuals. This is because the Software Installation Manual itself is actually the description of the post-condition that is desired. The system administrator will use the Software Installation Manual for Netscape if the required post-condition is that Netscape is installed.

- **Preparations**
  Some preparations must be performed before the actual installation can be performed. This section usually includes the step *"Log in as a user with sufficient rights"*.
- **Installation Procedure**
  The actual steps that must be performed in order to install the software.
- **Software Configuration**
  The actual steps that must be performed in order to configure the software so the user can use the software. These steps typically include setting the users name or registering the software with the vendor.
- **Finishing the installation**
  Here the steps are specified that must be performed after the installation has been completed. Typical steps in this chapter are the counter parts of the steps specified in the preparations chapter. If for example the preparations chapter specify to login as a specific user then this chapter will specify to log out again.
- **Installation Tests**
  To ensure the installed application can be tested this chapter contains test procedures with a description of the desired behaviour.
- **Cloning**
  A company usually purchases a larger number of identical systems in a single order to minimize the ordering costs. It is very inefficient to install each system individually using the available procedures. For example a single Windows XP installation with all standard applications [7] takes approximately a full working day. The practical way it is done is that a central installation is created of a single system with all company wide applications. The entire content of the hard disk of that system is duplicated ("cloned") onto the hard disk of the identical systems using the specialised tool called Ghost. After cloning when the new system is started for the first time some additional steps must be performed to make the system known in the network. This way a lot of time is saved making these systems available for the end users. In practise it is required for some applications to be stopped, cleaned or reset just before cloning and re-enabled just after cloning.

5.3.1.2 Installing Windows XP

The installation procedure for an operating system like Windows XP can be summarised as:

- Prepare BIOS for installation.
- Install Windows XP.
- Configure the network.
- Install Applications.

[7] The NLR has a clearly defined set of standard applications that must be installed on all workstations.

- Finish the Windows XP installation.
- Execute pre-cloning steps.

After the cloning of the installed system to an other system:
- Execute post-cloning steps.

For a standard desktop system with Windows a set of applications has been selected and for each of those applications a separate Software Installation Manual was written. During the step "Install Applications" the steps "Preparations", "Installation Procedure", "Software Configuration" and "Finishing the installation" of each of the required applications is executed. Note that the order in which the applications are installed can be an important factor in the obtained end result.

5.3.2 Types of knowledge and information

The types of knowledge that are available in the software installation manuals are all related to the state of the rest of the system at the time of installation and the behaviour of other applications after the installation has been completed.

The most obvious are the requirements that the software vendor places explicitly on the context where the application is to be installed. Three examples:
- Most software installations require a user with specific rights to be logged in during the installation of the software. On Linux systems this is usually the user called root [8], on Windows systems this is usually a user with Administrator rights.
- Microsoft Outlook 98 needs Internet Explorer (IE) version 4 or newer to be installed. IE in turn requires some version of Microsoft Windows to be installed [9].
- On the Linux platform the application called WorkPace [10] needs the "Record" feature of the XFree86 X-Server to be enabled.

[8] Or to be more precise: a user with user id 0

[9] Not entirely true: Some poorly supported ports to Solaris and HP-UX are available.

[10] WorkPace is an RSI prevention tool.

Some relationships between different applications are not so trivial. The strangest example I've seen so far is the reason why the current Software Installation Manual for Microsoft Internet Explorer contains the choice to install "Outlook Express". Initially this was not the case because the far more advanced application Outlook was installed and Outlook Express was therefore not needed. Many systems were installed without Outlook Express and there were no complaints from the users about this fact. About a year after the first systems were delivered with this setting, a security problem was discovered in Outlook and Microsoft made a fix available. This fix was distributed as part of a Service Pack for Internet Explorer. After this Service Pack was installed it was noticed that the problem had not been fixed. The problem turned out to be that the fix was not installed because Outlook Express had not been installed during the initial installation. This issue has led to the choice to install Outlook Express even though this application is really unwanted on the user's desktop.

5.3.3 Types of procedures and relations

The types of relations between the procedures found in the Software Installation Manuals are:

- The procedures have pre and post conditions which specify a dependency with other software being installed, specific settings in other software being set and a specific state the computer system is in before the procedure can be performed [11].
- Some procedures demand that other procedures are performed as well.
- Some procedures demand that other procedures are performed before or after the current procedure if they are performed at all.

5.3.4 Model requirements

The requirements for the model that come from this case are therefore:

1) A procedure must be able to specify a required initial state of the system before execution.
2) A procedure must be able to specify the state of the system after execution.
3) A procedure can demand the execution of a different procedure.
4) A procedure can demand a different procedure to be executed before or after if they are performed at all.

In these requirements the "procedure" is seen as an object that can perform a task (the actual procedure) and that can answer questions regarding the rules and constraints that define when and where this task can be performed. The answer to such a question may very well be dependent on the scope the procedure is in. Answering such a question does not necessarily require the task to be performed.

[11] For example the rights of the user account that is used during the installation.

5.4 The Vibration and Shock Test (VST) laboratory

On September 6th 2002 one day was spent at the Vibration and Shock Test (VST) laboratory at the NLR site in the Noordoostpolder. During this day several experts in this field were informally interviewed and some testing was observed.

5.4.1 Description

This section describes the Vibration and Shock Test (VST) laboratory and the procedures surrounding the tests.

5.4.1.1 Test facility

The Vibration and Shock Test (VST) laboratory is a test facility for determining the shock and/or vibration resistance of mainly avionic equipment and material samples.

The VST laboratory is an NLR facility where test samples can be placed onto a vibrating device called a shaker (this can viewed as a large loudspeaker) and optionally heat or cool the environment of the test sample.

Figure 2 The shaker testing a water boiler

The general idea behind the VST laboratory is to analyse how a device responds to the more dynamic mechanical environmental conditions, it will have to endure in the operational situation or during deployment.

Figure 3 The shaker in the horizontal configuration

The conditions that can be simulated using the facilities at the VST laboratory are:

- Accelerations between 1 g and 30 g
- Frequencies of up to 3000 Hz
- Temperatures between –80 ^{o}C and +200 ^{o}C
- Clean room: to provide a clean environment for space hardware

Figure 4 Nitrogen cooled test

Although the shaker can deliver about 20 kW of vibration energy, the devices tested are still limited in weight and the feasible accelerations and g-forces change with the weight of the device. For higher frequencies (up to 5 kHz) a small shaker is also available but the available power is only 3 kW.

The tests may consist of fixed or sweeping single frequency signals (sine testing) or of whole spectrum (random testing).
There are two kinds of tests that can be performed at this facility:

- **Vibration tests**
  In these tests the device is subjected to vibrations with frequencies and g-forces that regularly occur during normal use or during deployment. The requirement is that these vibrations do not damage the device. These tests are usually done on a prototype of the device and the vibrations during the entire lifetime are simulated in a short period of time.
- **Shock tests**
  In these tests the device is presented with shocks with g-forces that may occur during normal operation and during accidents and crashes. The shocks during normal operation shall not prevent the device from operating normally. Due to shocks during accidents and crashes, the device may cease functioning but it may not break apart or break loose from the fixture. If the device would break loose then it may become a projectile that could damage other systems or even people.

The kinds of devices that are tested vary from ovens and water boilers (see Figure 2 on page 21), which are to be installed in the galley of commercial airliners, to complete computer systems that are to be placed inside a satellite.
In the latter case, the vibrations from the launch are the major issue in the proper operation of the device and need to be tested before these devices are actually used. Also the new material called GLARE[12] that is to be used in the new Airbus A380 was tested at the VST lab.


Figure 5 Airbus A380

5.4.1.2 Procedures

The VST lab uses procedures that have been designed to document and streamline the communication between the customer and the test engineer. These procedures ensure that all the tests are performed that are required to ensure the behaviour of the device in the situation it will be deployed.

[12] Glare is a sandwich material of layers aluminium and glass fibre. The word 'Glare' stands for Glass fibre Reinforced Aluminium. Glare® was developed by the Delft University of Technology and the National Aerospace Laboratory (NLR).

These procedures have been implemented using forms. The first form “Formulier A” contains an overview of the customer, the project and the global requirements for the tests that are to be performed. The second form is “Formulier B” and is intended as a checklist to ensure that several required decisions are made correctly. This form has the following parts that also describe the order in which they should be performed during the actual test setup and the test runs:

B1: How the device should be attached to the shaker.

B2: Where should the control point sensor [13] be located?

B3: Determine the order in which the test runs can be performed in minimal time. This means optimising for minimal setup times between the test runs.

B4: Measure the resonance frequencies of the device. This is done by performing a sweep of all frequencies between 5 Hz and 2000 Hz at the very low force of 0.5 g. This will show where the device will resonate and will give a first impression of what the final result will look like.

B5: Measurement channels.

B6: Settings of the measurement processing equipment (a computer).

The tests are performed in the following steps:

1) The customer comes to the VST lab with a set of requirements for the tests that are to be performed on the device. If the customer does not have stated these requirements yet then the test specifications will be used from the guidelines described in section 5.4.2.3
2) ‘Formulier A’ and ‘Formulier B’ are filled to meet the customer’s requirements.
3) The tests are performed in accordance with ‘Formulier B’.
4) A test report with all the test results and conclusions is presented to the customer.

5.4.2 Types of knowledge and information

It was found that there are at least 3 fields of knowledge that are relevant to the VST laboratory.

1) The design and construction of the device.
2) The capabilities and side effects of the VST testing itself.
3) Operational knowledge on the situations in which various devices are to be installed.

The examples used in this section are the observations from the day at the VST laboratory.

[13] Used for reference measurement.

5.4.2.1 Design and construction of the device

One common problem that occurs is that the screws that are part of the device have not been fixed with a type of glue such as Loctite [14]. If no glue was used then the screws are likely to detach from the device in a matter of minutes and the device is likely to be damaged.

The device that was tested during the observation day was constructed roughly as shown in Figure 6. This device consisted of two boxes of which one was fitted inside the other. The two boxes were separated by blocks of a rubbery material (the red parts in Figure 6) for the purpose of obtaining the optimal temperature isolation of the contents of the inside box from the outside world.

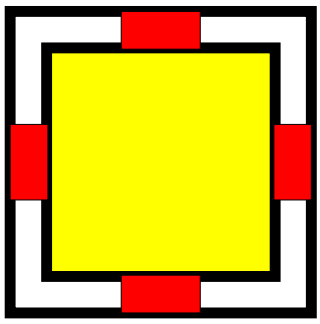

Figure 6 Diagram of the device

The problem with this design turned out to be that these rubbery blocks amplified the resonance frequency of the entire device so much that it was even nearly impossible to do the resonance sweep that is the first step in actual testing. The design and construction of the device caused the device to resonate violently even at the very low power setting of the resonance sweep. This meant that the shaker was unable to keep the accelerations within the specified range.

This simple example shows that the design choices made to accommodate the requirements of the application of the device conflicted with the environmental requirements of the device. This conflict may have been prevented if all of the requirements were known at design time.

The sensitivity of the device to electromagnetic pulses is a factor that may affect the testing procedure directly. When the shaker is started or stopped a very short high-energy magnetic pulse is emanated from the shaker. A device that for example contains a computer hard disk is destroyed when subjected to this kind of magnetic pulse.
The regular procedure for testing a device is:

Place device Ą Start shaker Ą Do tests Ą Stop shaker Ą Remove device.

This procedure minimises the power usage because shaker is started just before the actual test sequences and started directly after the tests have been completed. The procedure for these sensitive devices is slightly different:

Start shaker Ą Place device Ą Do tests Ą Remove device Ą Stop shaker.

This modified procedure forces the shaker to be powered during the mounting of the device onto the shaker but ensures the device is not subjected to the start/stop pulse.

[14] Loctite is a specific brand of glue for this purpose. http://www.loctite-europe.com/

5.4.2.2 Capabilities and side effects of the VST testing itself

The armature (the moving part) of the shaker that was used during the tests has a resonance frequency of itself at å2200 Hz. This implies that measurements near this frequency may be distorted by the behaviour of the shaker itself. This means that the measurement equipment itself is a factor in the measured results.

The shaker has limitations regarding the amount of power it can produce. This limits the heavier objects to lower g-forces.

5.4.2.3 Operational knowledge on the situations various devices are to be installed.

The customer who wants to have a device tested usually knows what kind of tests need to be performed. The customer is therefore able to provide the test engineer a specification of the g-forces and frequencies that the device must endure.

Some customers however do not specify this information but specify the type of environment the device must endure. For these customers actual measurements have been performed in the past in a significant number of environments. The findings of these measurements are available for specific groups of environments

- Civil aircraft applications: RTCA DO-160D ”Environmental Conditions and Test Procedures for Airborne Equipment”
- Defence applications: DoD: MIL-STD-810F “Test method standard for environmental engineering considerations and laboratory tests.”

Many customers use these references themselves to specify what test they need to have performed.

5.4.3 Types of procedures and relations

The procedures that have been found in the VST lab are very straightforward. Most of them are just checklists to ensure that all relevant questions are answered before the actual tests are done. The main problem is that these procedures are not augmented with the new knowledge that is discovered during actual testing. Some of these procedures are only retained in the head of the people that actually discovered the knowledge. This was regrettably the case with the magnetic start/stop pulse.

5.4.4 Model requirements

Only a few requirements follow from the VST lab case. The requirements that were found are:

1) Checklists must be possible.
2) The checklists must be easily updateable to ensure new ideas and checks are performed the next time a test is to be performed.

6 A model for procedural knowledge

This chapter describes the journey from requirements to model.

6.1 Requirements

In this section the requirements from sections 5.1, 5.2.4, 5.3.4 and 5.4.4 have been summarised.

Business requirements:

0.1) The model must have enough expressive power to represent all the procedural knowledge that was found in the provided cases.

0.2) The model must make it possible to filter the knowledge based on the actual requirements depending on the task at hand.

0.3) The model must allow version management and approval workflow of the knowledge.

0.4) A person without university training should be able to learn how to use the model. This means that the way in which the knowledge is represented should be intuitive to any user writing or examining a stored procedure.

The light bulb case:

1.1) It must be possible to reuse parts of a procedure in a different context.

1.2) It must be possible to model relationships between separate sub-procedures.

The Software Installation Manuals case:

2.1) A procedure must be able to specify a required initial state of the system before execution.

2.2) A procedure must be able to specify the state of the system after execution.

2.3) A procedure can demand the execution of a different procedure.

2.4) A procedure can demand a different procedure to be executed before or after if they are performed at all.

From the Vibration and Shock Test (VST) laboratory case:

3.1) Checklists must be possible.

3.2) The checklists must be easily updateable to ensure new ideas and checks are performed the next time a test is to be performed.

6.2 Representing Procedural knowledge

The only common representations for Procedural knowledge are:

- Procedural representation
- Declarative representation

Both representations will be described in more detail in the next two sections.

6.2.1 Procedural representation

Representing a procedure in a procedural form means that the procedure is represented as a flow chart with optional conditional branches (Figure 7). The procedural representation is also referred to as “imperative” [Ref. 7].

The procedural representation has the following advantages:

1) All choices in the procedures are defined explicitly.
2) The procedure is stored in a fixed form. This means that the procedures can be stored on a piece of paper in a form that is immediately ready for use (as shown in Figure 7).
3) Procedures can be nested into a hierarchy. This means that each step of a procedure can be a procedure in itself. This allows for reuse of procedures by referring to a procedure using a unique identifier. A very common example of this the name of a function within a specific scope as used in many computer-programming languages.

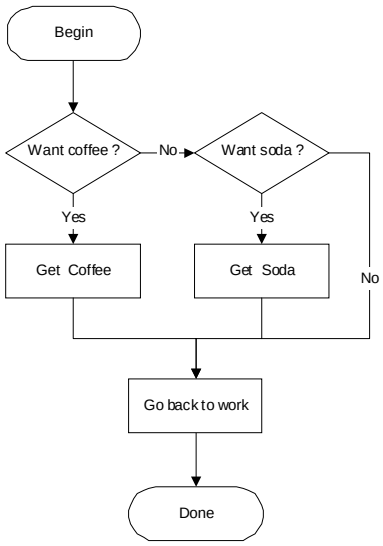

Figure 7 A simple flowchart

There are however also a few disadvantages to this representation.

1) The complexity of the flowchart becomes too high when the number of interrelated procedures increases.
2) The need for a specific step must be entered manually. If the a step needs to be used in several places, then this step must be inserted in all these places manually. This will lead to situations where a step is needed and unintentionally omitted.
3) It is very hard to maintain the procedures in this form due to this increased complexity.
4) The information on *why* a specific step is to be made is not stored. This knowledge is lost in this representation and needs to be stored separately.

This kind of representation is very common in expert systems ([Ref. 11] and [Ref. 12]).

6.2.2 Declarative representation

The declarative representation specifies a procedure in terms of a finite set of rules that implicitly determine the order of the steps needed to reach the specified goal. This means that the order in which these steps must be performed is determined by letting a (preferably automated) system determine an order of a subset of the available steps that leads to the required goal and obeys all the applicable rules. In computer science this concept has been extensively examined. Application programming languages like Prolog have been designed to use the declarative representation to code the tasks the application is to perform.

In "The Art of Prolog" [Ref. 8] a very simple example is given in only 2 steps in Prolog that determines the procedure for solving the well-known "Towers of Hanoi" puzzle [Ref. 9]. François Edouard Anatole Lucas invented this puzzle in the 19th century [Ref. 10].

Figure 8 Towers of Hanoi puzzle

Note that in a declarative representation procedures may have a parameterised relationship between the initial situation and the end situation. The tower of Hanoi example uses this parameterised representation to recursively use itself (with different parameters) and terminates this recursion on the second step.

Finding a solution means that the available steps are ordered based on the rules to form a procedure the get at the required goal. This is essentially deriving a procedural representation from the declarative representation for a specific transition. Finding such a solution is required because an average human cannot correctly follow all the required steps in the right order without the aid of a computer [15].

A simple example to illustrate this problem: the solution to N! [16] in two rules:

*N! = N * (N-1)!*

1! = 1

The procedure that is constructed to find the solution for 4!

*4! = 4 * (4-1)! Ą 3! = 3 * (3-1)! Ą 2! = 2 * (2-1)! Ą1! = 1*

Then substitute the answers:

*4! = 4 * (3 * (2 * (1))) = 4*3*2*1 = 24*

Note that the resulting procedure depends greatly on the task at hand. If the task were to calculate 6233167!, the procedure would be too large to perform by hand.

[15] Suggestion: Try doing the Towers of Hanoi with just 6 disks by hand as shown in Figure 8.

[16] N! = N * (N-1) * (N-2) * ... * 3 * 2 * 1

It is important to note a few issues about finding a solution using a declarative representation:

1) In some situations multiple valid solutions can be found.
   This means that there were several ways to get from A to B. The difference between the various solutions may be in the required cost, time or in the side effects that were not addressed in the rules. Depending on the actual situation the creator(s) of the procedures may decide that
   - more/different rules are needed to remove some of the produced solutions.
   - A score needs to be determined to decide which of the valid solutions is the best one.
   - all the presented solutions are equally good.

2) In some situations no valid solutions exist.
   This means that there is according to the available knowledge no way to get from A to B. Depending on the actual situation the creator(s) of the procedures may decide that
   - some rules need to be changed or removed.
   - more steps need to be added in order to make the procedure path from A to B possible.
   - there is indeed no way to get from A to B and nothing should be changed.

3) Finding a valid solution is an NP-complete [17] problem.
   This means that the time that is required to find a solution increases exponentially with the number of steps in the procedure. This practically means that if the number of steps becomes too large the time required may become several centuries using current computer systems. Increasing the computer power with a factor of 1000 (According to Moore's law [18] this would take about 15 years [19]) would not help if only a single extra step is added. This means that the only way to solve the NP-completeness issue is to design the model and the solutions finding algorithms to deal with it.

[17] NP-complete is sometimes referred to as NP-hard

[18] http://www.intel.com/research/silicon/mooreslaw.htm

[19] Double transistors every 18 months so a factor 1000 Ą 18 months * log(1000)/log(2) å 15 years

The declarative representation has the following advantages:

1) The declarative representation is more expressive than the procedural representation because for all the steps the exact context requirements are stored.
2) The procedure is stored as an unordered set of steps that are linked by rules. This means that adding a rule is very easy and has an impact in all the situations where the rule applies. The creator of a rule does not have to know exactly where the rule applies.
3) Storing knowledge in this form is often more intuitive than the procedural representation.
4) Complex procedures can in most cases be written with less 'lines of code'.

There are however also a few disadvantages to this representation.

1) The procedure is stored as an unordered set of steps that are linked by rules. This makes it very easy to create a rule that ensures that absolutely no solutions exist for specific situations. If a solution should exist then the author of the procedure must solve a very difficult puzzle: find the 'bug' in the rules.
2) The procedure is stored as an unordered set of steps that are linked by rules. This means that using the procedures requires solving a difficult puzzle which is NP-complete.
3) It is very hard to get an overview of the consequences of adding, changing or deleting a rule.
4) Many people will have problems actually using the declarative representation even though it is very intuitive in itself. The problem is that many people try to find a structure in order to get an overview of the procedures they are working with. Because this is an NP complete problem they will fail in this effort very quickly. This is a part where some training of the authors and a helpful user interface are essential.

6.3 Atomic procedure

In order to ensure that procedures can be modelled the concept of *atomic procedure* is introduced.
First attempt for a definition:

An atomic procedure is a procedure that cannot be split into smaller sub-procedures.

If examined closely then the conclusion must be drawn that an atomic procedure that strictly follows this definition is not practical. Take for example the very small procedure *"Push the power button"*. This procedure can be described as an atomic procedure because it describes a step in the procedure that cannot be broken down into smaller steps. But is this really the case? The fact is that this procedure can be broken down into something like this:

- Locate the power button
- Place your finger against the power button

- Increase pressure of finger against power button until you feel a click.
- Decrease pressure of finger against power button until you feel no more pressure.
- Remove finger from power button.

Each of these steps can be broken into even smaller steps that may involve the neurons being fired in the finger of the person performing the procedure. It is obvious that describing a procedure in atomic steps that are defined at this level is useless for business applications.

So for this thesis we will use the following definition:

An atomic procedure is a procedure that cannot be split into smaller sub-procedures that still contain useful information.

This definition is not very sharp and makes a few important assumptions. This definition means that if the procedure says *"Push the power button"* that the user of the procedure is assumed to know how to push a button and is able to locate the 'power' button. This definition therefore assumes the user has mastered a set of basic procedures. Writing procedures with this definition in mind has a very important side effect: The writer of a procedure must assume or explicitly state the required knowledge of the person performing the task.

This means that if for example a Software Installation Manual (section 5.3) contains the procedure *"Click OK"*. Because the people using the Software Installation Manuals are assumed to be systems administrators with a more than basic level of computer skills, this procedure is considered to be atomic. The procedure *"Click OK"* contains enough detail regarding the actual task at hand for the person who has to execute it.

A person doing a beginners course in using a computer will need a lot more detail than an experienced systems administrator performing the same task. The same procedure *"Click OK"* may have to be written as something as elaborate as this:

Move the mouse (the little box next to your computer keyboard that fits in your hand and that contains 1,2 or 3 buttons and an optional wheel) over the table (don't lift it from the table) until the arrow or hand image on the screen is on top of the graphical representation of a button on the computer screen that has been labelled with the text "OK". After this has been done the left mouse button must be pressed and released without moving the mouse over the table. Note that if the mouse has been configured to be left-handed then the right mouse button must be clicked instead.

6.4 Designing the model

6.4.1 Transitions in a state space

As mentioned in section 3.2 a procedure can be considered as a description of

"What to do to get from situation A to situation B?"

This really means that a procedure is a description on how to get from a specific state ('A') into a different state ('B'). Each state can be described as specific values or ranges of values for a set of variables. A state can therefore be represented as a point in an N-dimensional state space where N is the total number of used variables. Using that representation a procedure is a set of possible transitions within that state space.

Each variable that is a dimension in the state space can be of a different type. The possible types that are useful for procedures are:

- Boolean: true or false.
- Enumerated: a set of discrete values. These values can also be pieces of text.
- Integer: an infinite set of discrete numerical values.
- Real: a continuous range of numerical values.

Each of these variables can have an additional 'state' in addition to the specified values. This is the 'undefined' state. Later on we will show that this is needed to ensure requirement 2.4 can be met.

Note that other mathematical number groups like Q (fractions like ¾) and C (complex numbers like 4 + 2i) can all be represented by using two vectors of the types Integer and/or Real.

From the perspective of the procedure the problem exists that the number of variables N of the N-dimensional space is unknown. It's even worse: in real life situations the number of variables changes with every update of the available procedures. The practical solution is that a procedure should only address the dimensions it has an effect on. Therefore a procedure does not have to be a translation of a point to another point within the N-dimensional space but will most likely be the description of the translation of a hyper plane into a different hyper plane.

In general a procedure therefore consists of:

1) Initial state requirements (pre-conditions).
2) Procedure that perform the actual translation.
3) End state (post-conditions).

The initial state requirements should only describe the requirements on the variables that have a direct effect on the procedure itself. For each variable this may be a specific value or a range of values (if possible).
The end state should only specify the value of the dimensions that have been changed. Note that a variable can have an end state without being part of the initial state requirement. Example: the procedure *"Paint wall with white paint"* does not require the wall to be in any particular colour before hand, after this procedure has been completed the variable *"Wall colour"* will have the value *"White"*. Note that this description of the requirements of a procedure is really a declarative representation that describes how this procedure fits into the 'environment'.

For a full example of the requirements the procedure *"Place new light bulb"* from the case in section 5.2 is examined:
The initial state requirements are:
- New light bulb available.
- Lamp housing is reachable.
- Lamp housing is open.
- Lamp housing is empty (there is no light bulb present).

After the procedure has been completed the changed variables are:
- New light bulb is not available.
- Lamp housing is not empty.
- Light is working. [20]

Note that these variables have not changed
- Lamp housing is reachable.
- Lamp housing is open.

6.4.2 Related procedures

When examined closely this simple model does not meet all requirements that have been stated in section 6.1. The most obvious omissions are requirements 1.2 and 2.3: related procedures. Let's assume all procedures of the replacing a light bulb procedure have been represented in a declarative form. When determining the procedural representation that is to be used to place the new light bulb the steps to finding this procedural representation are something like this:

1. Start with an empty collection of procedures to use
2. Requirement "Light is working."
   Ą Add "Place new light bulb" procedure

[20] Let's just assume that nothing else can possibly be wrong, so the light must be working now.

3. Requirement “Housing is open” for “Place new light bulb”
   Ą Add “Open Housing” before “Place new light bulb”
4. Requirement “Lamp housing is empty”
   Ą Add “Remove defective light bulb” before “Place new light bulb”
5. Requirement “Housing is open” for “Remove defective light bulb”
   Ą Add “Open Housing” before “Remove defective light bulb”
6. All requirements are met Ą Sort the procedures and present end result.

The procedure “Close Housing” wasn’t a requirement anywhere so it wasn’t added to the set of procedures that is to be used. The only correct way to ensure that this procedure is added is by adding a requirement that the “Housing must be closed” at the end of the procedure.

The easiest way to solve this is to add an additional state requirement: the required end state of a variable. This means that at the end of the entire procedure the state of a variable must be a specific value or it must be defined as an invariant [21].

It is not possible to just assume that variable changes caused by any automatically added procedures must be returned to their respective initial state. An example where this is not wanted is the installation of Outlook, to install Outlook you need a new version of Internet Explorer. After Outlook has been completed the Internet Explorer needs to remain available. This means that it is important that the requirement if a variable must be reset or not, is explicitly specified.

6.4.3 Choosing a representation

Regarding the choice for a representation we are faced with the following dilemma:

- If the provided procedures are described in a *pure procedural form* then the complexity of the procedures becomes extremely difficult to handle. If for example a step needs to be added for a specific case, then in all parts of the workflow where this case might occur this procedure must be inserted. The risk is that the person updating the procedures forgets to update a part of the procedure simply because the number of steps has risen to an unmanageable scale.
- If the provided procedures are described in a *pure declarative form* then the amount of work required to writing the rules for each atomic procedure that is to be documented becomes too large and the time required to find the needed procedure may take too long. On the other hand atomic procedures that are required in many situations now only have to be written once.

[21] Invariant: a variable that hasn’t changed when comparing the before and after situations.

To resolve this dilemma a good look at the definitions of procedures is in order. In section 3.2 the relation between procedures and sub-procedures was defined. An important issue about this definition is that it does not specify *how* the relationship between the sub-procedures has been represented. This can be either procedural or declarative (section 6.2). The implication of this is that it is possible to mix both procedural and declarative representations at different levels in the hierarchy of related procedures. An example on how this can be achieved using the example from section 5.3.1.2 is visualised in Figure 9.

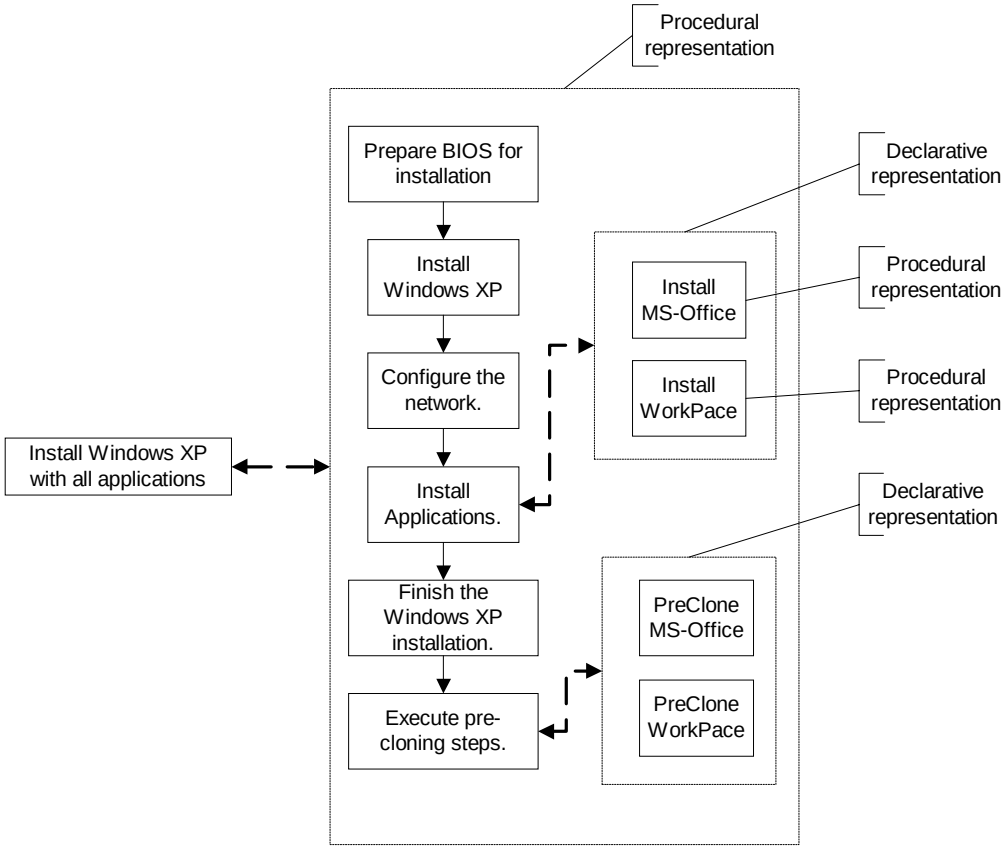

Figure 9 Mixing representations.

In this example the procedure “Install Workpace” can be placed before or after the procedure “Install MS-Office”, but it cannot be placed before “Configure the network” or after “Finish the Windows XP installation”. By defining different representations at different locations it is possible to ensure that all kinds of procedural representations are possible. This means that choosing a representation is not done at the level of designing the model but at the level of the actual implementation of a specific set of procedures.

It is important to note that the references to procedures do not imply exclusive usage of these procedures. To allow the reuse of the same procedures in a different context each procedure must be referenced by some unique identifier.

Using this unique identifier it is possible to create logical groups of procedures that are related in some way. A logical group makes it possible to get an overview of all procedures directly related to for example WorkPace

Declarative procedures are capable of producing more end states than requested at a specific moment. Because of the choice for a hybrid procedural/declarative representation also the procedural procedures have this capability if one of the sub-procedures is declarative. This means that the end state of a procedure cannot be determined beforehand. If the top-level requirements change then the end state will change as well.

6.5 Final model

In this section the final model is described in a more formal way.

The general form of this model is a tree structure where a procedure is a node in the tree. Each node can be either an atomic procedure or a set of procedures that are linked together either in a procedural or in a declarative way. All procedures are always referred to using a globally unique identifier.

The notation that is used:

<name> ::= <spec>	The object <name> is defined as <spec>
<name1> | <name2>	The object consists of either <name1> or <name2> [22].
<name1> & <name2>	The object consists of <name1> and <name2>.
*<name> **	0 or more instances of <name>
<name> +	1 or more instances of <name>
{ }	used for grouping expressions
NULL	The indicator for “Undefined” / “Unused” / “Empty”.

[22] This is an ‘exclusive or’.

The final model for procedural knowledge we arrive at is the following:

Expression			*Meaning*
Why	::=	{ A description of why this is defined this way. } | NULL	This is a descriptive attribute used in many places that describes the knowledge why a specific value or setting is set. This attribute is always optional.
atomic procedure	::=	Usable description of a procedure.	An atomic procedure as defined in section 6.2.
variable state	::=	variable name & variable value & { Why }	Defines a single variable state.
variable name	::=	Unique identifier.	The globally unique identifier (name) of a variable.
variable value	::=	NULL | { {value} | { range of values } } +	The value, values or range of values a variable has. Note that the values can be used to reference the value of other variables in the definition of this procedure.
procedure	::=	{ procedure name } & { declarative set of procedures | procedural set of procedures | atomic procedure } & { initial state } & { end state } & { final state } & { must before } & { must after } & { Why }	The definition of a generic procedure.
procedure name	::=	Unique identifier	The globally unique identifier (name) of a procedure.
declarative set of procedures	::=	procedure name +	An unordered collection of procedures
procedural set of procedures	::=	{ procedure name + } & { explicit workflow for the specified procedures }	A collection of procedures ordered into a workflow.

Expression			*Meaning*
Variable state expression	::=	Logical expression of { variable state * } & { Why }	This is a logical expression that allows for logical IF...THEN…ELSE… , AND, OR, NOT, and XOR operators of variables.
initial state	::=	Variable state expression	The initial state that must be met before a procedure can begin.
end state	::=	Variable state expression	The end state that contains the changed variables after the procedure has been completed.
final state	::=	Variable state expression	These define the values of specific variables that must be true at the end of the procedure at the top of the hierarchy.
must before	::=	procedure name *	A list of procedures that must be executed before this procedure if they are executed.
must after	::=	procedure name *	A list of procedures that must be executed before this procedure if they are executed.
Auto trigger procedure	::=	{ procedure name } & { trigger state } & { insert before | insert after } & { Why }	A procedure that is added directly before or after the procedure that causes the specified trigger state.
trigger state	::=	Variable state expression	These define the values of specific variables that trigger the insertion of an auto trigger procedure.

Note that it is possible to model the various relations between the procedures differently than done in the above model. The reason for this very straightforward representation is that it is much easier to understand and much easier to use (requirement 0.4).

6.6 Representing atomic procedures

Now that a model for procedural knowledge has been established an important issue remains: How do we represent the procedure description of an atomic procedure or a small procedural procedure. If the system is to generate the documentation for the user in a form that is specific for this user then a static representation like the Word97 of PDF documents, as described in the problem description (section 2.1), won't work. A representation for documentation that seems to be very appealing is XML. With XML it is possible to represent hierarchical information using plain text. Many tools exist to manipulate XML files and convert the end result into other file formats like PDF, RTF and HTML. [Ref. 13]

6.7 Merging

If a procedure and all of the sub-procedures (i.e. an entire branch of the tree of procedures) have been represented in a pure procedural form then the option exists to represent this procedure as a single very large 'atomic' procedure. This merging of procedures should be used with caution because the system can no longer reuse sub parts of the specified procedure in a different context. The advantage however is that a procedure can be managed more easily and be edited with more conventional editing tools.

7 Case analysis

In this chapter the requirements from the various cases will be checked to see if all them can actually be modelled in the model proposed in Chapter 6. For each of the examined cases some of the possible issues of building an actual implementation have been described.

7.1 Business requirements

Requirement 0.1: *The model must have enough expressive power to represent all the procedural knowledge that was found in the provided cases.*

Requirement 0.2: *The model must make it possible to filter the knowledge based on the actual requirements of the user.*
The model lets the user specify the required begin and end state for which a procedure is required.

Requirement 0.3: *The model must allow version management and approval workflow of the knowledge.*
Each procedure has a unique identifier. The system used to manage the procedures can be equipped with a version management and workflow features. The part of the system that is used to determine the procedure can then be fed with just the procedures of a specific version.

Requirement 0.4: *A person without university training should be able to learn how to use the model. This means that the way in which the knowledge is represented should be intuitive to any user writing or examining a stored procedure.*
The model has been designed to be a clear as possible. If this is clear enough for practical purposes is hard to determine. The fact remains that the author of the procedures needs to understand the procedures that are documented very well.

7.2 The light bulb case

7.2.1 Requirements

These are all the requirements from the light bulb case described in section 5.2.

Requirement 1.1: *It must be possible to reuse parts of a procedure in a different context.*
All procedures have a unique identifier that can be referenced in many different contexts. If these procedures are part of a declarative procedure then this can even be done automatically.

Requirement 1.2: *It must be possible to model separate sub-procedures that are related.*
Section 6.4.2 has a full description on how this has been implemented.

7.2.2 Building an implementation

Although this is just a fictitious case it is still useful to examine the issues regarding an implementation. The NEN electrical norms contain most of the rules regarding electrical installations. The main problem with these rules is the sheer number of rules that is described. Because these rules change from time to time it would be a very inefficient use of resources for a company if they would enter all the rules in a system themselves. The most efficient location for entering these rules into any automated system would be the original author: the Nederlands Normalisatie Instituut.

7.3 The Software Installation Manuals case

7.3.1 Requirements

These are all the requirements from the Software Installation Manuals case described in section 5.3.

Requirement 2.1: *A procedure must be able to specify a required initial state of the system before execution.*
This is an explicit part of the model (section 6.5).

Requirement 2.2: *A procedure must be able to specify the state of the system after execution.*
This is an explicit part of the model (section 6.5).

Requirement 2.3: *A procedure can demand the execution of a different procedure.*
Section 6.4.2 has a full description on how this has been implemented.

Requirement 2.4: *A procedure can demand a different procedure to be executed before or after if they are performed at all.*
This is an explicit part of the model (section 6.5).

7.3.2 Building an implementation

For the Software Installation Manuals the initial state can be stored into a database that contains the exact set of procedures (including the exact version numbers) that have been executed using the initial installation of the system. This way if an additional piece of software needs to be installed the system is capable of producing the exact documentation required for this task with minimal questions for the system administrator performing this task.

7.4 The Vibration and Shock Test (VST) laboratory case

7.4.1 Requirements

These are all the requirements from the Vibration and Shock Test (VST) laboratory case described in section 5.4.

Requirement 3.1: *Checklists must be possible.*

A checklist is nothing more than a linear workflow of several atomic procedures of the form: "Check that … is …".

Requirement 3.2: *The checklists must be easily updateable to ensure new ideas and checks are performed the next time a test is to be performed.*

This is essentially the same requirement as requirement 0.3.

7.4.2 Building an implementation

The manuals used to determine which test are to be performed can be placed into the described model. A procedure could be created that described the test procedure with specific frequencies and other settings with the end-state "Tested for rocket launch". Note that the problems regarding the electromagnetic pulse can be described in the procedure by having the various steps reordered depending on the variable "Device is sensitive to EM pulses".

8 Conclusions and further research

In the present study a model for representing procedural knowledge has been designed that combines the features of both declarative and procedural representations. Because of this mixture a single implementation can be constructed that allows the author to choose a different representation for different parts of a procedure. This additional flexibility results in a wider range of practical applications than the examined existing tools have shown.

In computer science only the procedural and declarative representations are widely used. This implies that apparently there is no practical need for an additional representation for procedural knowledge in this context. Based on this apparent lack of any other representations, we can conclude that the model designed in this paper is capable of representing all common procedures, because both common representations have been combined.

To what extent this model is more practically applicable is a point for further research. This question has nothing to do with what the model can represent, but with what the author of a procedure is capable of modelling.

This is, however just the model, not a working system. When actually implementing this model in a specific environment additional research is required. Even before such an implementation can be constructed the following questions still need to be answered:

- How should the required the level of detail be determined for an atomic procedure ?
- How should the best representation for the various parts of a set of procedures be determined ?
- How should an automated system that employs this model interact with the user ?

The next step is to implement the model in an automated system that allows the user to query the procedure that this specific user needs in this specific situation. The idea is to make this system in such a way that it will ask the user for the value of a variable if the value is unknown. Implementing the model is an issue that will require additional study to ensure that the ease of use is maintained and provided to the end users.

Many issues still need to be resolved before this model can actually be implemented in an automated system. Some of these issues are:

- Constructing a complete procedure based on the requirements specified by the user is an NP-complete problem. Is it possible to design an algorithm to solve this problem within an acceptable time for practical problems?

- How should the system interact with the user in order to make it as user friendly as possible?
- Can some of the initial states be placed into a database to reduce the number of questions the user needs to answer?

9 References

[Ref. 1] E-Learning as Vehicle for Knowledge Management
R. Choenni, R. Walker, R. Bakker, W. Baets,
In Proc. SOL 2001 2nd Int. Workshop on Supporting Organizational Learning,
Tokyo, Japan, October 20-22

[Ref. 2] Website by Deepak Kumar
http://blackcat.brynmawr.edu/~dkumar/UGAI/kr.html#paradigms

[Ref. 3] The Knowledge in Knowledge Management
Fred Nickols
http://home.att.net/~nickols/Knowledge_in_KM.htm

[Ref. 4] Technical documentation: An integrated Architecture for Supporting the Author in Generating and Resource Editing.,
Springer 2002
LNAI 2443, pp 122-131, 2002
Nestor Miliaev, Alison Cawsey and Greg Michaelson

[Ref. 5] Mavim website: http://www.mavim.com

[Ref. 6] NEN electrical norms (http://www.nen.nl) created by Nederlands Normalisatie Instituut (http://www.nni.nl)

[Ref. 7] Structure and Interpretation of Computer Programs,
MIT Press 1985
Harold Abelson et al.
ISBN: 0-262-01077-1

[Ref. 8] The Art of Prolog,
MIT press 1986.
Leon Sterling & Ehud Shapiro,
ISBN: 0-262-19250-0

[Ref. 9] Websites regarding the “Towers of Hanoi” puzzle:
http://directory.google.com/Top/Science/Math/Recreations/Famous_Problems/Tower_of_Hanoi/

[Ref. 10] Websites regarding the creator of the “Towers of Hanoi” puzzle:
http://www-groups.dcs.st-and.ac.uk/~history/Mathematicians/Lucas.html
http://www2.evansville.edu/ck6/bstud/lucas.html

[Ref. 11] Expert Systems, Principles and Programming,
PWS-Kent Publishing 1989
Joseph Giarratano
ISBN: 0-87835-335-6

[Ref. 12] Kennissystemen,
Addison Wesley 1992
Luc Steels
ISBN: 90-6789-327-7

[Ref. 13] Various websites with information regarding SGML, XML and the tools to convert documents into other formats:

The XML Cover Pages	http://www.oasis-open.org/cover/sgml-xml.html
XML Tutorial	http://www.w3schools.com/xml
XML FAQ	http://www.ucc.ie/xml
SGML History	http://www.oasis-open.org/cover/sgmlhist0.html
XML Standard	http://www.w3.org/XML
XML Software	http://www.xmlsoftware.com
	http://www.xml.org
	http://www.xml.com
Docbook	http://www.oasis-open.org/docbook
	http://docbook.org
DSSSL	http://www.oasis-open.org/cover/dsssl.html
	XSL http://www.oasis-open.org/cover/xsl.html