

SimaPro 7

Wizard Manual



product ecology
consultants

SimaPro 7 Wizard Manual

**Introduction into
programming of Wizards in SimaPro 7**

March 2006

Colophon

Title: Wizard Manual
Introduction into programming of Wizards in SimaPro 7

Written by: PRé Consultants
Suzanne Effting

Report version: 1.2
Date: March 2006
Language: English
Availability: PDF file

Copyright: © 2004-2006 PRé Consultants. All rights reserved.

PRé Consultants grants the right to distribute and print the unchanged PDF version of this manual for non commercial purposes only.

Parts of the manual may be reproduced in other work only after permission and if a clear reference is made that PRé Consultants is the author.

Support: phone +31 33 4555022
fax +31 33 4555024
e-mail support@pre.nl
web site www.pre.nl

Contents

1	INTRODUCTION	1
1.1	INTRODUCTION OF WIZARDS	1
1.2	STRUCTURE OF THIS MANUAL	2
1.3	GETTING STARTED WITH WIZARDS.....	2
1.3.1	The principle of Wizard writing	2
1.3.2	Where to place a wizard	2
1.3.3	Distribution of wizards.....	3
1.3.4	Wizards in the SimaPro explorer	3
1.3.5	What the user sees	4
2	THE WIZARD CONCEPT	6
2.1	NODES: BUILDING BLOCKS OF THE WIZARD	6
2.1.1	Overview of nodes.....	6
2.1.2	General properties	7
2.1.3	Message Node	8
2.1.4	Enter Name Node	9
2.1.5	Enter values node	10
2.1.6	Select process/product stage node	11
2.1.7	Choose node.....	12
2.1.8	Call node.....	12
2.1.9	Calculate node.....	13
2.1.10	Operations node	14
2.1.11	Show chart/table node	16
2.1.12	Show LCI results node.....	17
2.1.13	Show tree node	18
2.2	VARIABLES	19
2.2.1	Product stages Variable or Process Variable.....	19
2.2.2	Product stage or Process REFERENCE variables	19
2.2.3	Text and numeric variable	19
2.2.4	Results variable	19
2.2.5	Linking information to variables.....	19
2.3	NODES AND VARIABLES	20
2.3.1	Relation between nodes and variables	20
2.3.2	Text and numeric parameters in texts	21
2.4	PROGRAMMING WIZARDS.....	23
2.4.1	Debugging.....	24
2.4.2	Wizard structure.....	25
2.4.3	Other things you have to know....	26
3	EXAMPLES AND PRACTICE	28
3.1	EXAMPLE: ASSEMBLY	28
3.2	EXAMPLE: CALCULATIONS.....	31
3.3	BOTTLE PRACTICE	32
3.3.1	Wizard programming assignments.....	32
3.3.2	The result: what the user sees	37
4	PRODUCT SYSTEMS.....	38

1 Introduction

1.1 Introduction to wizards

Since 1998, PRé Consultants has developed an alternative way of presenting the LCA results using wizards. Wizards present themselves as a series of questions to the user.

Wizards are usually focused on a specific product. In practice this means Wizards are used for groups of users that have to deal with a similar topic or problem. There are three main applications for wizards:

1. **LCA by non-LCA experts.** Wizards can be used to develop an easy user interface for people with no, or limited knowledge of LCA methodology. The questions can focus on the functional unit and some basic assumptions. By answering the questions, the user can perform its own LCA, without actually having to understand all the details of LCA methodology, or how SimaPro works (see Figure 1).
2. **Publishing an LCA.** Wizards can be used to show the results of your LCA study to interested parties. Wizards can replace a static LCA report, and give the audience an interactive view of your LCA. The presentation of the results is transparent and you can even allow users to perform their own sensitivity analysis and make comparisons from several alternatives. The Guided Tour you see when you start the demo provides an example of such a use of wizards. Although the user only sees the questions and the results of the LCA, underneath the wizard the full SimaPro database and all the intermediate results are available for further inspection; the results are thus completely transparent.
3. **Wizard for LCA.** Wizards can be used to create a wizard for LCA modeling that is repetitive. Wizards can help to decrease the difficulties in life cycle modeling. Especially with multiple users working a large LCA project, wizards can guarantee that the same methodology for life cycle modeling is followed by all SimaPro users. An example of a wizard to decrease the difficulties in LCA modeling is the LCA wizard in the standard database of SimaPro.

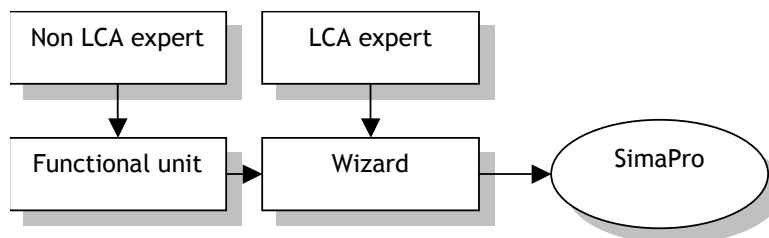


Figure 1 The LCA expert can develop an interface between the non-expert user and the tool. The expert is responsible for developing the interface and the database in such a way that the results are methodologically correct. The user is responsible for submitting correct data on the product composition and the life cycle.

There are two types of roles involved with wizards:

- The Wizard developer. This person should have sufficient understanding of the LCA methodology.
- The Wizard user. This person does not have to understand the intricacies of LCA methodology. His or her main role is inputting data on the way the product is made and on its life cycle.

LCA experts who have a thorough understanding of SimaPro can develop wizards. The LCA expert can assume the responsibility that the Wizard results in a methodologically correct LCA, while the user can take the responsibility that the product and life cycle are properly defined. Any user of a SimaPro {Developer, right!?!} or Multi-user version can program wizards.

1.2 Structure of this manual

Wizard writing is something you learn by doing. This manual can be used as a reference when learning how to program wizards. Before you can start developing your own wizards, some knowledge about wizards is required. Therefore a short explanation is provided here of the contents of this manual.

Wizards are built using several building blocks, called nodes. Chapter 2 starts with an overview these building blocks, followed by a more detailed explanation of their use and function. In wizards, variables are used to store the information created by the wizard. Section 2.2 gives an overview of the all variables available for wizard writing, and in section 2.3 the relation between variables and nodes is explained. Section 2.4, Programming wizards, gives an introduction in wizard programming: linking nodes in a smart way. Chapter 3 shows a variety of wizard examples and gives the possibility to practice wizard making on your own.

Before wizards writing is explained in chapters 2 & 3, section 1.3 provides information important to know before you start writing wizards.

We advise you to take a look at the LCA wizard in the SimaPro standard database, to see some of the explanations in this manual 'live' in SimaPro.

1.3 Getting started with wizards

1.3.1 The principle of Wizard writing

Creating wizards is not a task for the novice user. You will need a good understanding of LCA and of the way SimaPro works, before you start creating your own wizard.

Before you start programming wizards, you should address the following questions:

- Who is the target audience that is going to use your wizard?
- What skill level do they have, how much do they know about the product and LCA?
- Which type of product systems should they be able to model, and which characteristics of the product or the life cycle should be open for modification?
- What is the availability and quality of the data?

Wizard are usually combined with a specific datasheet, so often you can only start programming the wizards if you have at least a rough version of the data and impact assessment methods ready.

1.3.2 Where to place a wizard

Most wizards refer to processes or product stages that are already in the database.

Before you start writing the wizard, at least some of this data should already be available. Wizards and variables are stored in a specific project or library. The data (assemblies, processes, etc.) created or edited from within the wizard are stored in the project that is opened while running the wizard (Figure 2 A and B).

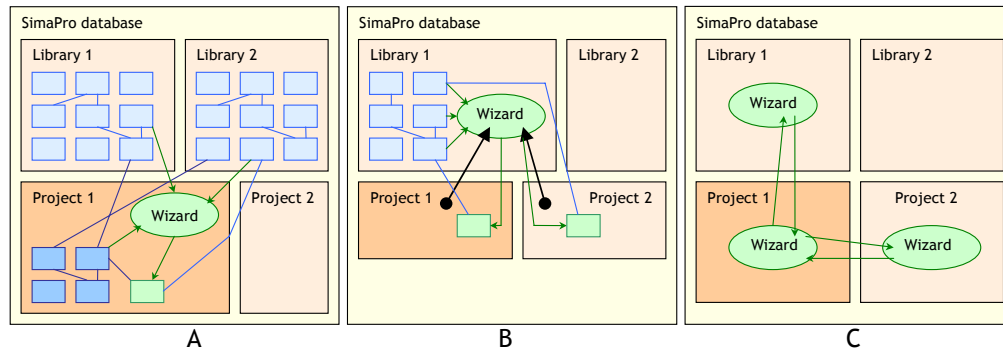


Figure 2 Graphical overview of some of the possibilities to store a wizard in SimaPro.
 A: The wizard is stored in a project, and can use data from all libraries. Items created by the wizard are stored in the project where the wizard is located. These new items created by the wizard can have links to other items in the project, or links to library items. This wizard cannot be used in other projects.
 B: Wizard stored in a library. This wizard can be used in the library itself, but also in all projects. Items created by the wizard are stored in the project that was opened when running the wizard.
 C. Wizards that do not create or edit items can jump to wizards in other projects or libraries

You can either store a wizard in a library or in a project. Libraries cannot refer to other libraries; therefore a wizard in a library can only call data from within that same library. Therefore, in cases where your wizard will use data from several libraries, you need to place the wizard in a project. A wizard stored in a normal project can use data from within the project and from all libraries available (see also Figure 2 A and B).

In case you make a wizard for demonstration purposes (i.e. a wizard that does not produce or edit any data), it is possible to let wizards jump from within a project to a wizard in another project or library (Figure 2 C).

The items created by a wizard in a library are stored in the project that was open before the user started executing the wizard, and/or in projects opened while running the wizard.

1.3.3 Distribution of wizards

Wizards are developed in a Developer or in a Multi-user version of SimaPro. However, wizards can be distributed to the users in a so-called “Light” version of SimaPro. This version has the benefit that the libraries cannot be changed. So if you store your data in a library, the wizard will always be able to find the data you assumed to be available in the wizard.

After presenting the first version of the wizard, there will probably be a reason to provide updates. Sending a new version of the libraries and the wizards to the users can do this. Users who want to maintain the data generated with the previous version can import this data from their old database into their new database.

1.3.4 Wizards in the SimaPro explorer

You can select the Wizards item on the *LCA explorer* index, to get an overview of ALL wizards in the currently open project and in the libraries (see Figure 3). This list contains both executable and non-executable wizards (see section 1.3.5). In this Wizards section of SimaPro, new wizards can be developed. The wizard developer can create new folders under ‘Wizard’ to store the wizards.

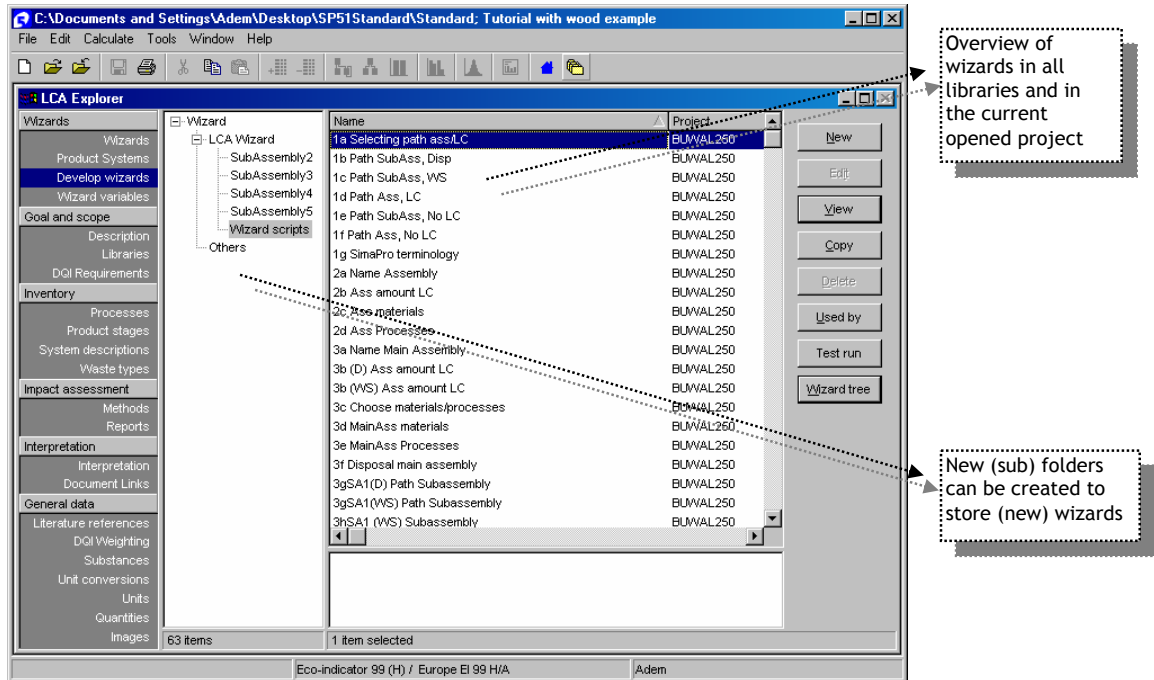


Figure 3 Wizard section in the explorer window

1.3.5 What the user sees

SimaPro users can find available wizards in the LCA explorer index, under 'Wizards'. The wizard page shows a list of executable wizards in the currently open project and in the libraries (see

Figure 4).

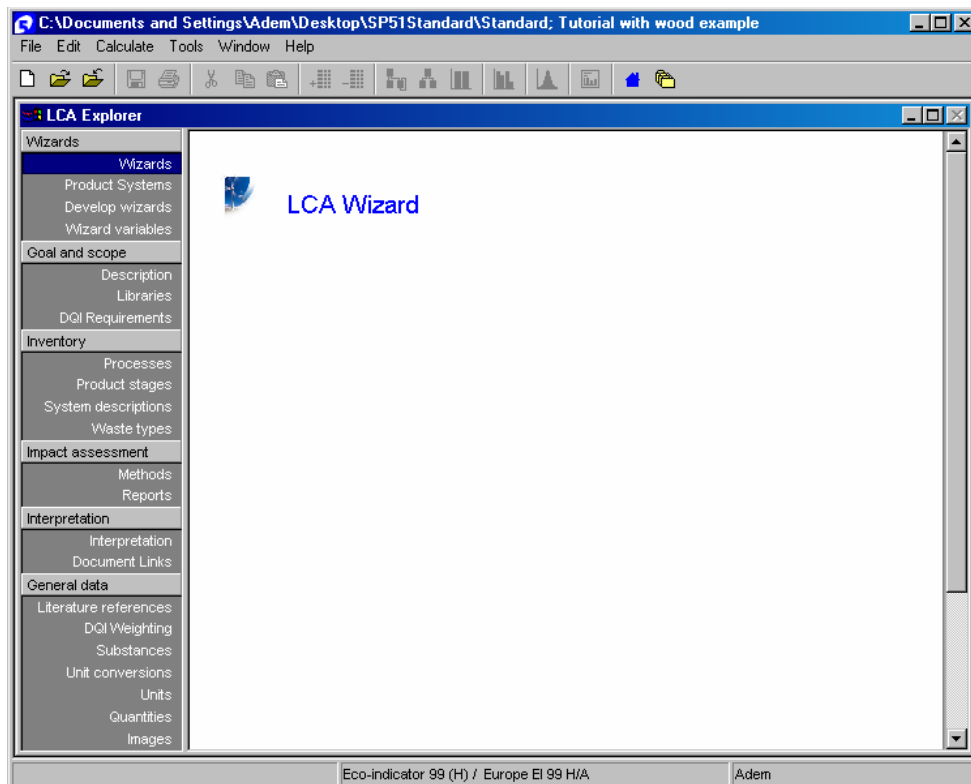


Figure 4 SimaPro users can find available wizards in the LCA Explorer index, under 'Wizards'

Executable and non-executable wizards

It is possible to create a set of wizards where each performs a specific task. This makes it possible to structure your work and develop wizards in a modular way. In order to help the Wizard user find out which wizard he has to start, you can label one or more wizards as "executable". Only these wizards will be shown to the user, other wizards will not be shown. In the wizard section of the explorer, where wizards are stored and are available for editing, all wizards are visible (see Figure 3).

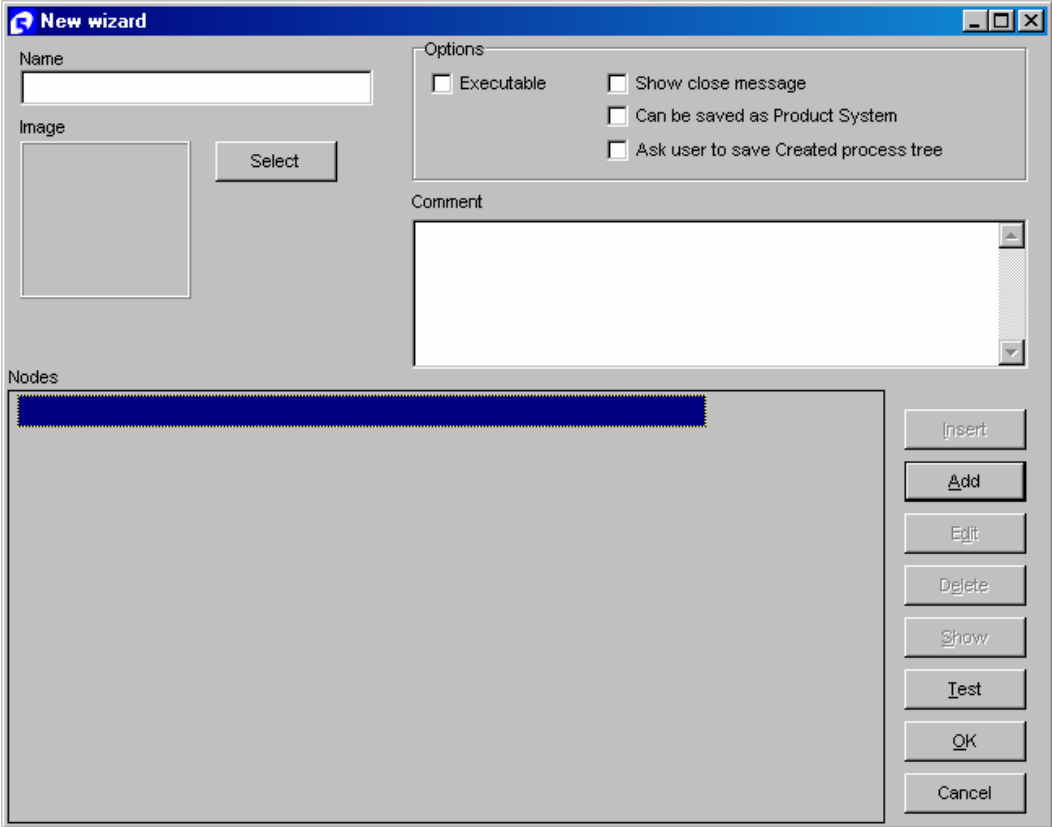


Figure 5 Partial view of window creating a new wizard. Under 'Options' there is the possibility to make the wizard 'Executable'. Only executable wizards are visible in the LCA Explorer index, under 'Wizards'.

Only for executable wizards is the text field under 'Comment' visible to the user. In non-executable wizards, you can use this field to make notes useful to yourself as the wizard designer, such as a description of the content of the wizard.

2 The wizard concept

2.1 Nodes: building blocks of the wizard

2.1.1 Overview of nodes

Wizards are built as a series of nodes that are placed in a certain order. A node represents a basic operation, such as a window with a question. The main task in wizard programming is defining nodes. Most nodes are quite easy to define, although there are also some more complex nodes. We distinguish three specific groups of nodes: for data entry, results display, and processing. There is also one general node: the message node.



Message





Message node

Displays a text message to the user.

Data entry nodes



In data entry nodes, the wizard receives information from the user. There are four different data entry nodes. Each of these nodes has the possibility to show text to the user. This can be used to give information or clarify what the wizard user should do.

The choose node is the most abstract data entry node, since wizards allow for any kind of choices.

 Enter name	Enter name node Allows the user to enter a name; for example the name of an assembly.
 Enter values	Enter values node Allows the user to enter a value; for example the amount of steel in an assembly.
 Select process/product stage	Select process node Allows the user to select a process into an assembly; for example, he can choose between the different types of steel in the database.
 Choose	Choose wizard route node Allows the user to decide which is the next task.

Processing nodes (invisible nodes)

Processing nodes are not visible to the user of wizards. These allow the wizard to perform actions in the background.

 Call	Call wizard node Jumps to and activates another wizard.
 Calculate	Calculate node Calculates the results of one or more processes or product stages with an impact assessment method.



Operations

Operations node

This is one of the most complex and advanced nodes. Here you can create new processes and product stages. It allows you to include arithmetic operations within the wizard. For example, you ask the user for the height and width, and you let the operation node calculate the surface.

Result nodes

These nodes display results to the user.

 Show inventory	<p>Show inventory node Lets SimaPro show the Life cycle inventory results or results of impact assessment per substance.</p>
 Show impact assessment	<p>Show impact assessment node Lets SimaPro show the results of impact assessment in a predefined way.</p>
 Show network	<p>Show network node Lets SimaPro display a graphical process network representation.</p>

2.1.2 General properties

You can either insert or add a new node in a wizard using the buttons on the right side of the wizard window (see Figure 5). ‘Insert’ puts the new node above the selected node; ‘Add’ places the new node below the selected node. Nodes cannot be moved, so be careful where the new node should be placed.

In the next sections, the nodes are explained in more detail. Some ‘fields’ appear in several nodes, therefore we will explain these first in Figure 6, using the window of an ‘Enter values node’ as an example.



Enter values

The 'description' field is used to give a name to the node, and is not visible to the wizard user. The description field can take up to 50 characters, and should be used to give an indication of what the node does.

The 'text' can be used to enter text that will be displayed to the wizard user. You can use this to give information or to clarify what the user should do

Wizards use Variables to store information. Variables themselves are never visible to the wizard user. The names you use for variables should identify their purpose (see section 2.2)

Figure 6 Explanation of general fields in node windows

2.1.3 Message Node

The *Message* node is used to display messages in a wizard. It is very simple and consists of a text window, description field and three buttons.

In the description field you define the name of the node. The wizard user will not see it, but you will need it to be able to find which node does what.

The text field can contain flat text and string (%s) or value parameters (%d). Use the String parameter button to link this parameter to a string or value variable (see section 2.3.2).

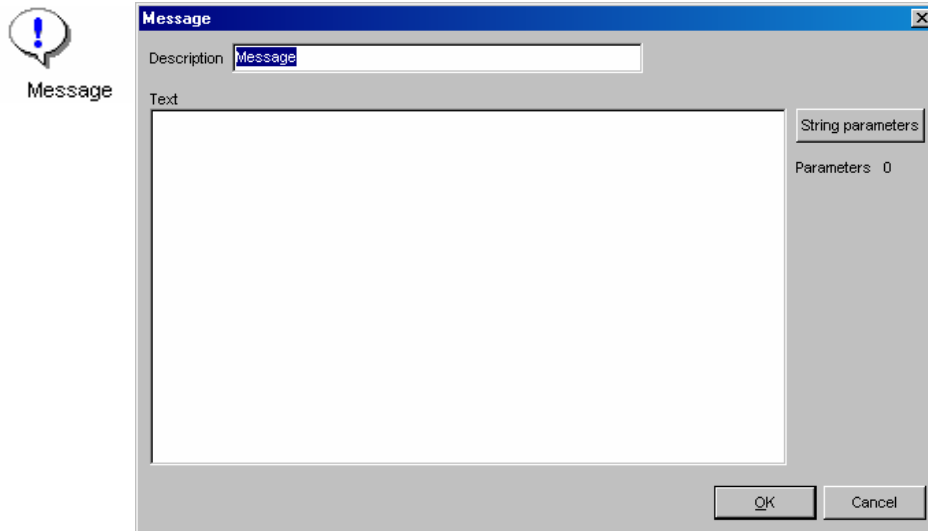


Figure 7 Window of a message node. The text in the white field under 'Text' is displayed to the user of a wizard

2.1.4 Enter Name Node

This node is used to ask the user to specify a name of an object. This name is assigned to a variable, so it can be used in the next wizards. You can have the node check whether the name is already in use, to avoid double names. You can define the error message that is displayed in that situation. Under 'Label' you enter the text that is displayed to the user for the field where he or she enters the name.

The screenshot shows the 'Enter name' wizard window. It includes a 'Description' field with 'Enter name', a large 'Text' area, a 'Variable' field with a 'Select' button, a 'Label' field, a 'Default Name' field, a 'Check name on existing' dropdown menu, and an 'Error message' field. Callouts provide detailed explanations for these fields.

Callout 1 (top right): The variable field displays the variable. The text (string) typed by the user is assigned to this variable. You can use the 'Select' button to select one of the existing variables or create a new one. The variable of an enter name node is always a text variable (see section 2.3)

Callout 2 (left): The Label field appears on the wizard screen just before the window in which the user types in the 'name'. For example, you explain in the text field why you would like to know a name of a product. In the label you simply state 'product name'

Callout 3 (right): In the 'default name' you can type a suggested name. The user can accept the suggestion, or overwrite it with a new name.

Callout 4 (bottom): You can use this option if you want to find out if the name already occurs in the database. Depending on your specification it will check, and generate a message that you can define in the Error message field.

Figure 8 Window of an Enter name node. The text in the red lines is displayed to the wizard user.

2.1.5 Enter values node

This node is used to ask the user to specify one or more values (numbers). These numbers are assigned to value variables, so they can be used in other nodes.

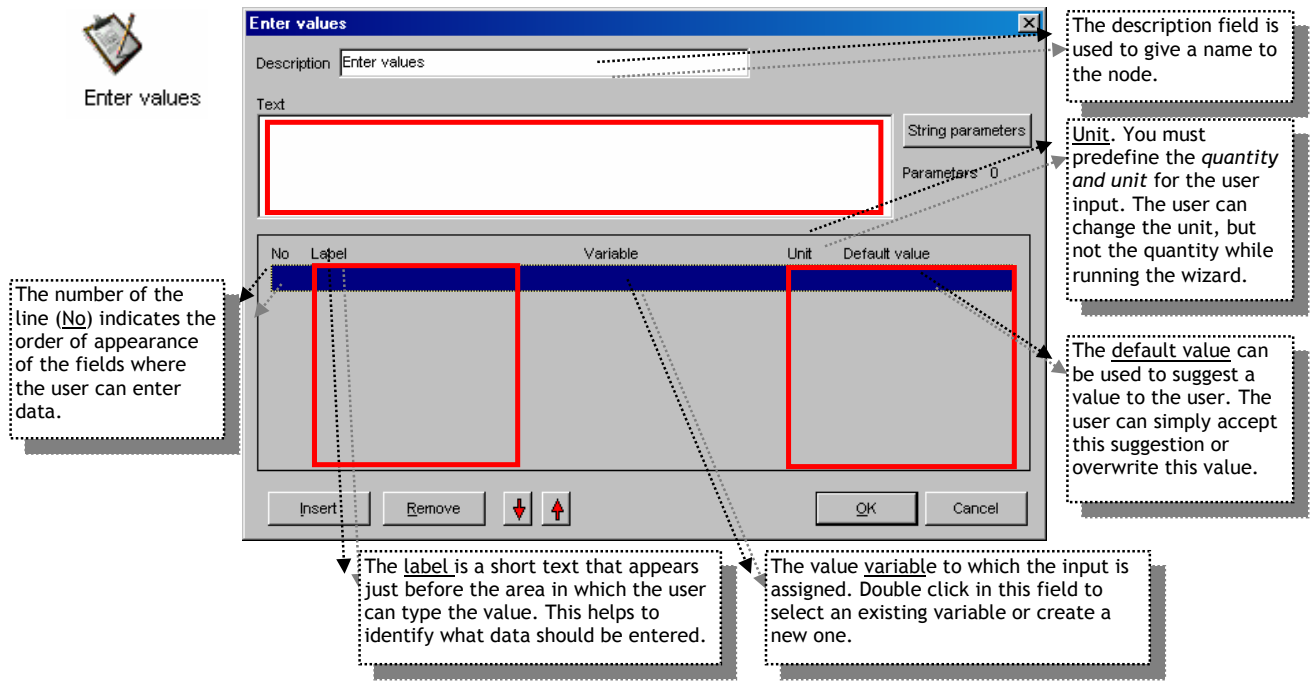


Figure 9 Window of an Enter values node. The areas within the red blocks are visible for the wizard user

You can specify up to six fields in one 'enter values' node where the user can enter data. The quantities of these fields do not need to be the same in one node.

2.1.6 Select process/product stage node

This node is used to let the user choose from a number of predefined processes or product stages in the database. The selected item is assigned to a *process or product stage REFERENCE variable*. See also section 2.2.

Example: Suppose you would like to let the user select the most appropriate transport process. List in this node the most relevant data on truck and rail transport. The user can select from this list and add the transport amount in an ‘enter values’ node.

Use the ‘Add’ button to select the relevant processes or product stages, from which you would like to let the user make a choice. If you choose to select a ‘category’, the wizard will show all processes or product stages in this category; otherwise, you can select individual processes or product stages.

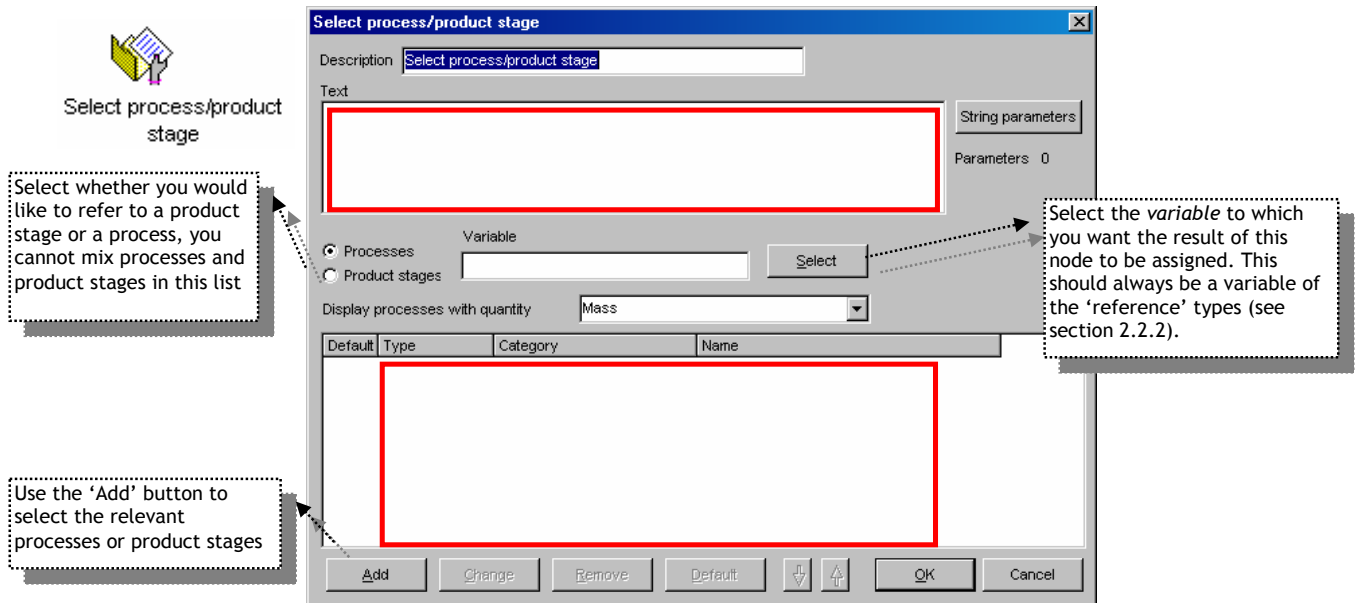


Figure 10 Window of a select process/product stage node.

If the wizard you are building is stored in a library, you will only be able to select processes that are in this library. If your wizard is built in a project, you can select any process or product stage in this project and in all libraries that are linked to your project. This limitation is needed to avoid cross-references between libraries (see also section 1.3.2 Where to place a wizard).

You cannot specify an amount or unit in this ‘select process/product stage’ node, as the unit of the process or product stage will be used. You collect information on the amount via an *enter values* or *from an operations* node.

2.1.7 Choose node

This node is used to let the user choose between two or more alternative routes through the branches of the wizards. You can only add a choose node at the end of a wizard.

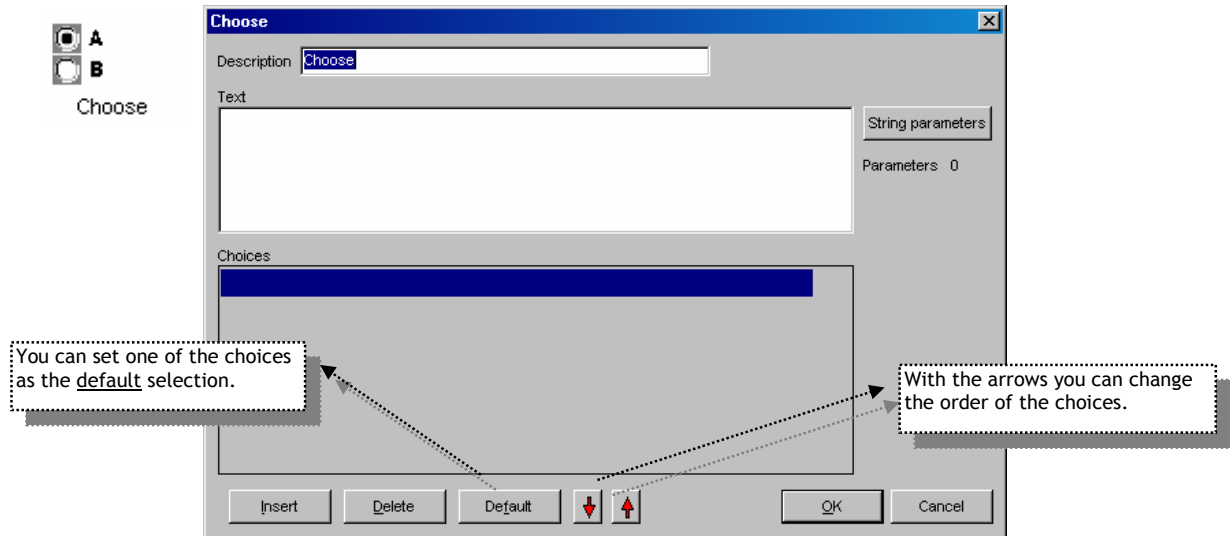


Figure 11 Window of a choose node.

Under 'Choices' you can type one line to describe each choice.

The choose node will only generate a number of alternative branches, without specifying what each branch will do. To define what will happen when a choice is made, close the Choose wizard node editor. You will see that a number of dummy branches have been created in the Wizard overview screen.

Double click on these branches and choose the next node that is to be executed when the user makes a choice. For each choice you have to define at least one node. If you do not fill in the choices, you will not be able to run the wizard, and an error will appear.

Suppose you want to end a choice and branch immediately to another wizard (see section 2.4.2 Wizard structure). In that case, you can add an operation node. This operation node can be left empty and will not perform any task.

Due to a programming limitation you cannot change the first type of node belonging to a choice. The workaround is simple: define a new choice and link this choice to the node you want to have executed and delete the old choice. However, if you delete a choice, all the following nodes of that branch will be deleted as well. Nodes cannot be moved or copied! To make the wizard development flexible, you can use *call wizard* after a choice (see sections 2.1.8 and 2.4.2).

2.1.8 Call node

The Call node is a node that is not visible to the user. It is used to start a sub wizard from within another wizard. After the sub wizard (and its possible sub-sub wizards) has been executed the program will return to this node and execute the next node.

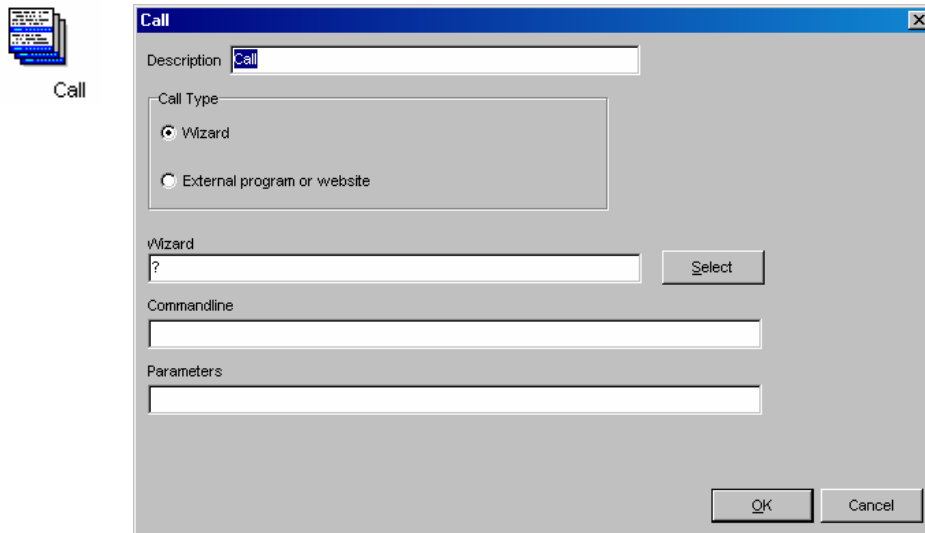


Figure 12 Window of a call wizard node

In this node you specify which wizard is to be started. With the ‘Select’ button, you can choose to call a wizard in the current project or libraries. A wizard can also call itself! See section ‘2.4.2 Wizard structure’ to learn how you can use this node to design a looping wizard structure.

2.1.9 Calculate node

This node is not visible to the user. Its purpose is to analyze or compare processes or product stages, and assign the full results (LCI scores as well as impact assessment results) to a result *variable*.

1 + 1

Calculate

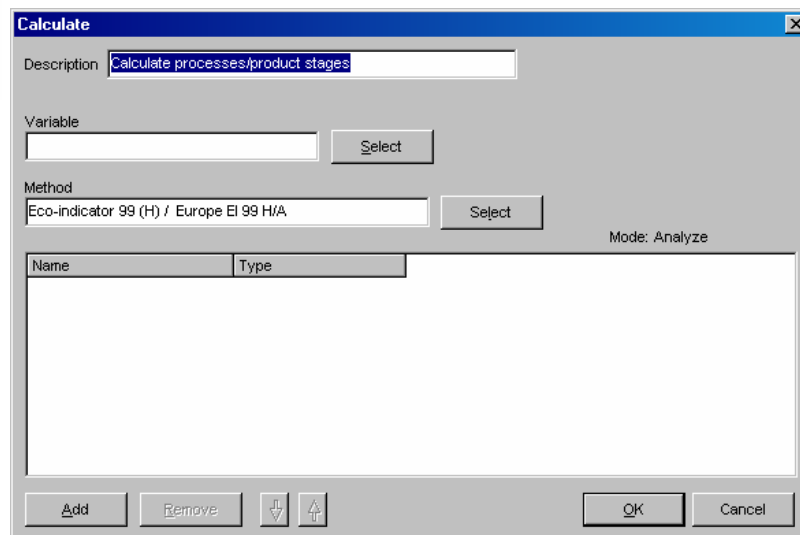


Figure 13 Window of a calculate node

The calculation node is defined by selecting one or more process or product stage, or process or product stage references, and by selecting an impact assessment method and the result variable that is to be assigned to the result. If you only select one process or product stage variable, the calculation will run in analyze mode, if you select multiple items, it will automatically switch to compare mode.

2.1.10 Operations node

The operations node can be considered the core of the wizard. The operations node allows you to perform arithmetic operations with variables, and to store the results of these operations in other variables. You can also create new process stages and processes (with some limitations). This node is by far the most complex and comprehensive node. You need to understand the concept of *variables* before you attempt to define operations (see sections 2.2 and 2.3).

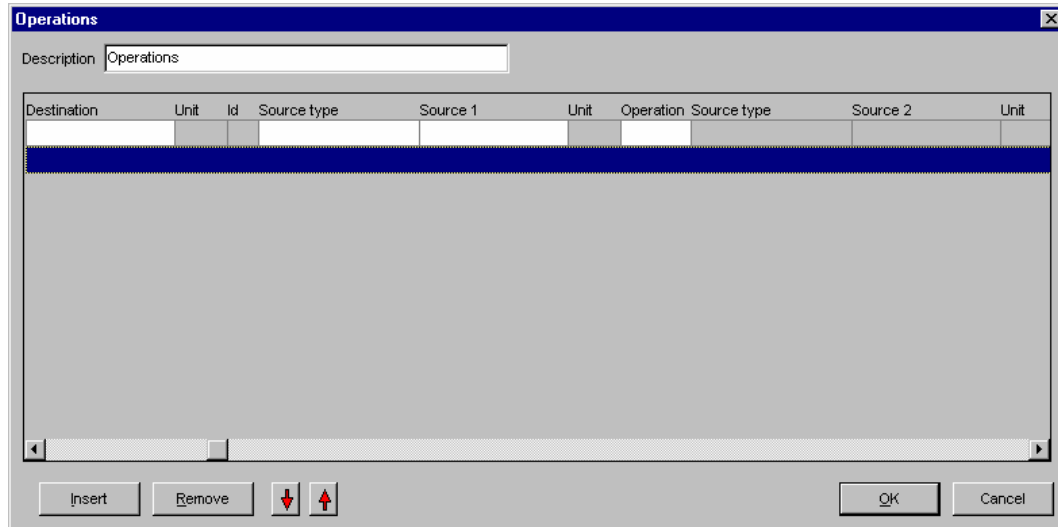


Figure 14 Window of the operations node.

The operations node runs in the background: the user will not directly see this node. The core of the Operations node is a table, in which you specify the wizard operations in the following way:

You create 'it' by doing 'something' with a 'thing'.

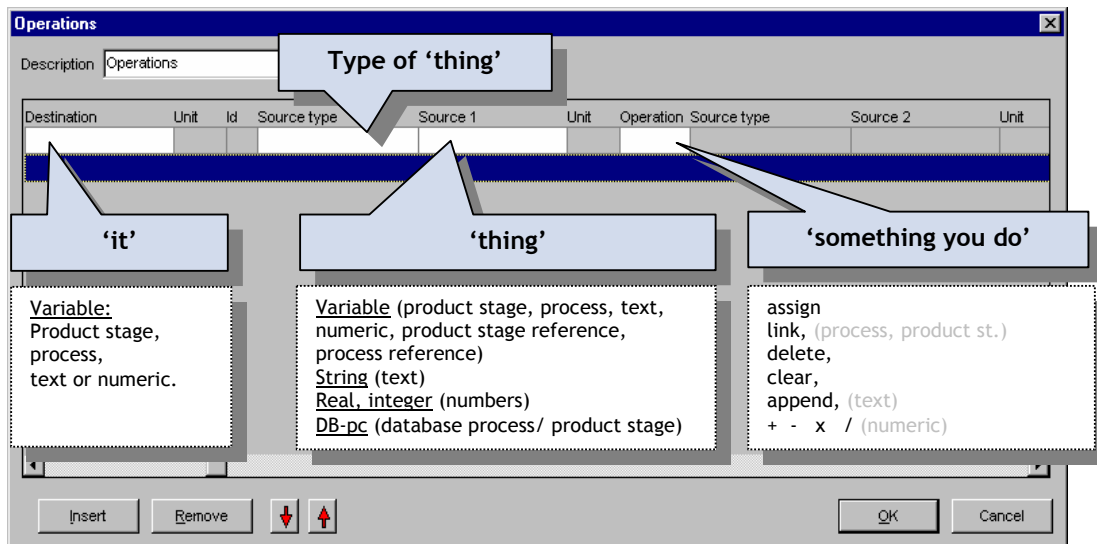


Figure 15 Operations: Create 'it' by doing 'something' with a 'thing'

Figure 15 illustrates the meaning of this cryptic explanation. Under 'Destination' you specify a variable where the wizard stores the result of the action of that line. Under source 1 you specify the 'thing' that is the subject of the action, and under 'Operation' you specify 'something that the wizard will do with that thing.'

The source type always accompanies the sources. This is the type of variable that is used as a source.

You should use 'Assign' when you want your destination ('it') to become source 1 ('thing'). Assign is used to give product stages and processes their name, quantity and value. 'Link' is used to attach variables (like process references etc.). For example, a product stage life cycle links to a product stage assembly.

There are a few other fields in operations that become available depending on the operation you enter. These fields are 'Unit' and 'Id'.

Unit

If the destination variable is the result of the calculation of two source variables, the result will be expressed in the standard unit. If you want to express the result in a different unit, double click this field to open the *Select a unit* dialog Product stage (see Unit field (Select a value)). You must check if the result can be expressed in the unit you select. An error message is displayed if a unit is selected for a different quantity from that of the result of the calculation.

Id

If the destination variable is a *Product stage* or *Process* and the source type is numeric (*Value*, *Integer* or *Real*), the *Id* determines to which part of the Product stage or process the data in the source variable are assigned. The source variable can contain a numeric value and a unit.

The *Id* value is interpreted as follows:

- Id =-1** This *Id* value can only be used with a *Process* variable. The data in the source variable will be assigned to the output product of the process. For example if you program a processing process in wizard, you use 'Id=-1=1kg' to indicate that all the data in the process are applicable on processing of 1 kg product. Note that this is the *first* output product because wizards cannot create multiple output processes.
- Id=0** The data in the source variable will be assigned to the *Value* and *Unit* properties of the destination variable. This is the default *Id* value.
- Id>0** The data in the source variable will be assigned to another source variable, which has been *linked* to the destination variable (a process or a product stage). The *Id* value relates to the position of the linked object, i.e., the first, second, third, etc. (see Figure 16).

Destination	Unit	Id	Source type	Source 1	Unit	Operation	Source type
thermos jug			String	Thermos jug for mode		Assign	
thermos jug			DB-pc	Glass (white) B250		Link	
thermos jug		1	Real	0,2	kg	Assign	
thermos jug			DB-pc	PP granulate average		Link	
thermos jug		2	Real	0,2	kg	Assign	
thermos jug			DB-pc	Injection moulding		Link	
thermos jug		3	Real	0,2	kg	Assign	
thermos jug			DB-pc	Heat gas B250		Link	
thermos jug		4	Real	4	MJ	Assign	
thermos jug			DB-pc	Truck 28t B250		Link	
thermos jug		5	Real	0,4	tkm	Assign	

Figure 16 Example of an operation to create a thermos jug assembly with links to processes in the database. In this example Id>0 is used to link values to processes in the assembly

With other destination variable types (*Product stage-ref*, *Pc-ref* and *Value*), the data in the source variable are always assigned to the *Value* and *Unit* properties of the destination variable. In other words, although the *Id* field is disabled, it is automatically given a value of 0.

The **Insert** button is used to add a new *destination* variable. Clicking this button, or double-clicking the space beneath an entry, will open the *Select a variable* dialog.

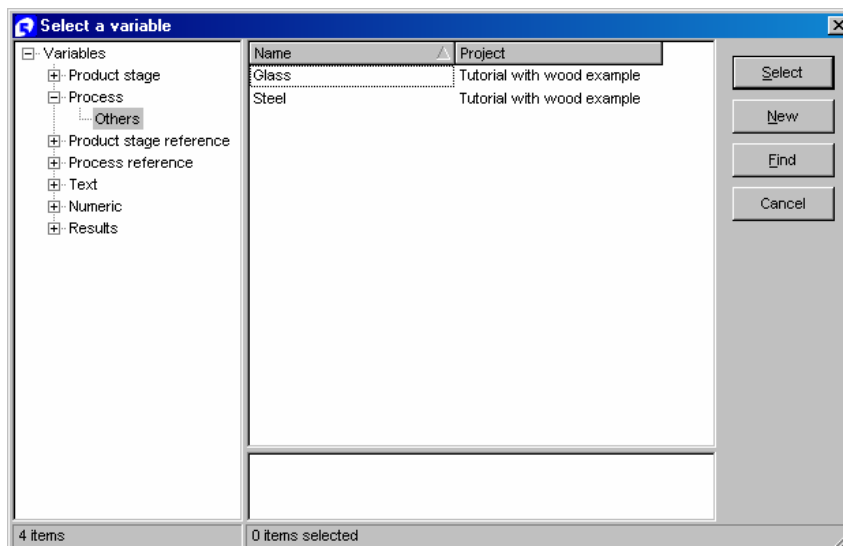


Figure 17 'Select a variable' window to select a variable in an 'Operations' node

See section '3.1 Examples and Practice' for examples of how to use the operation node.

2.1.11 Show chart/table node

The Show chart/table node is used to display the contents of a result variable. Such a result variable is filled with data from a *calculation node*. This means that you can only use the graph node if you have used a calculation node first. In the show chart/table node you cannot choose the impact assessment method. This is done in the calculation node.

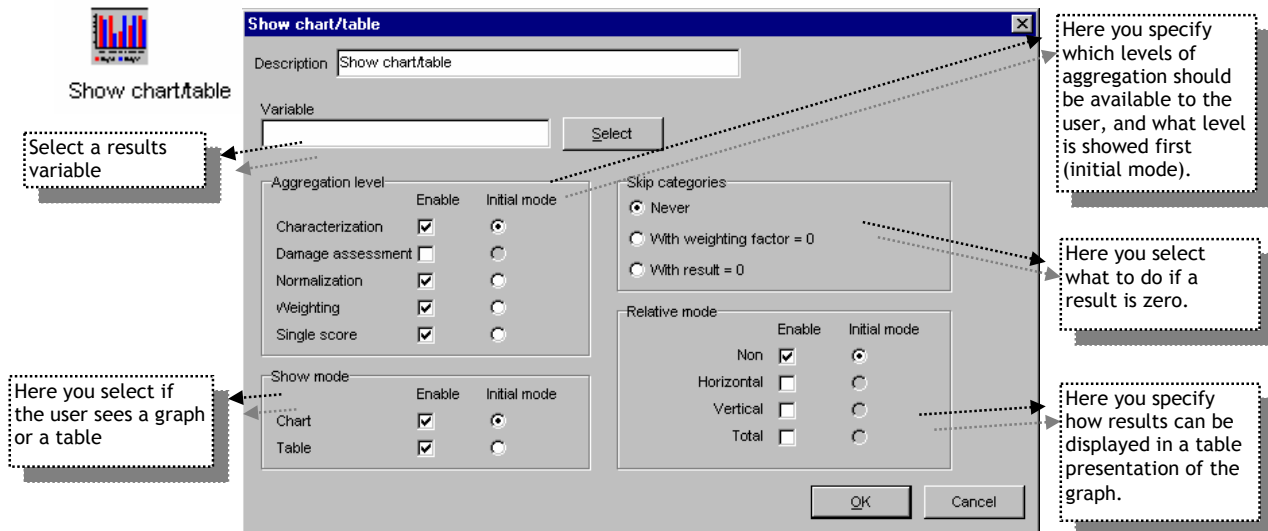


Figure 18 Window of a show chart/table node

You define the graph node by specifying which result variable must be presented, and how this must be done. How refers both the definition of options available to the user, and the way the graph is displayed initially.

You can specify which levels of aggregation should be available to the user. For example, you may want to prevent the user from using weighting. You can also specify the initial level of aggregation. For example, you can specify characterization as the initial level. The user will see that the characterization level is shown first. If you have permitted the user, he or she may choose to use one of the other aggregation levels. Under relative mode you specify how results can be displayed in a table presentation of the graph. The following options are available:

- Non refers to displaying a normal table.
- Horizontal refers to a table that shows the relative contribution to an LCIA result over multiple columns.
- Vertical refers to a table that shows the relative contribution as percentages per column.
- All refers to a table that shows the relative contribution in percentages over the whole matrix. This means only one cell in all rows and columns gets a 100% score; the rest are relative to this one cell.

2.1.12 Show LCI results node

The LCI results node is used to display the contents of a result variable as an LCI result table. Such a result variable is filled with data by a *calculation node*. This means that you can only use the LCI results node if you have used a calculation node. The results displayed by the LCI results node are of course dependent upon the impact assessment method defined in the calculation node.

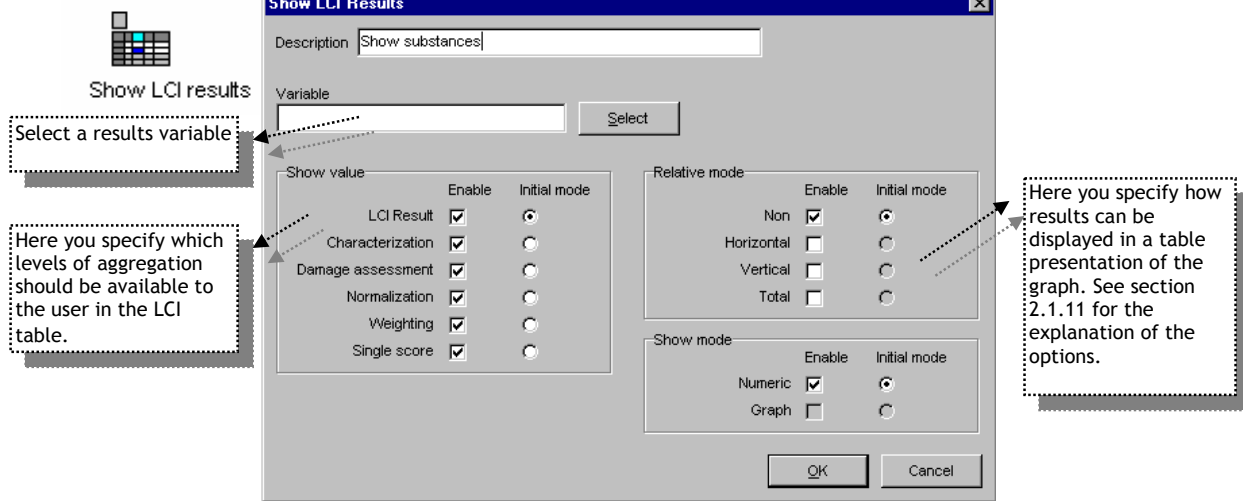


Figure 19 Window of a Show LCI results node

You define the Show LCI results node by specifying which result variable must be presented, and how this must be done. How refers both the definition of options available to the user, and the way the table is displayed initially.

You can specify which levels of aggregation should be available to the user in the LCI table. For example, you may want to avoid that the user uses weighting. You can also specify the initial level of aggregation. For example, you can specify characterization as the initial level. The user will see that the characterization level is shown first. If you have permitted the user, he or she may choose to use another aggregation level.

2.1.13 Show tree node

The show tree node is used to display results in a tree. In contrast to the *graph node* and the *substance node* the results for the tree are calculated within the *tree node*. Thus, *tree nodes* do not need to be preceded by a *calculation node*.

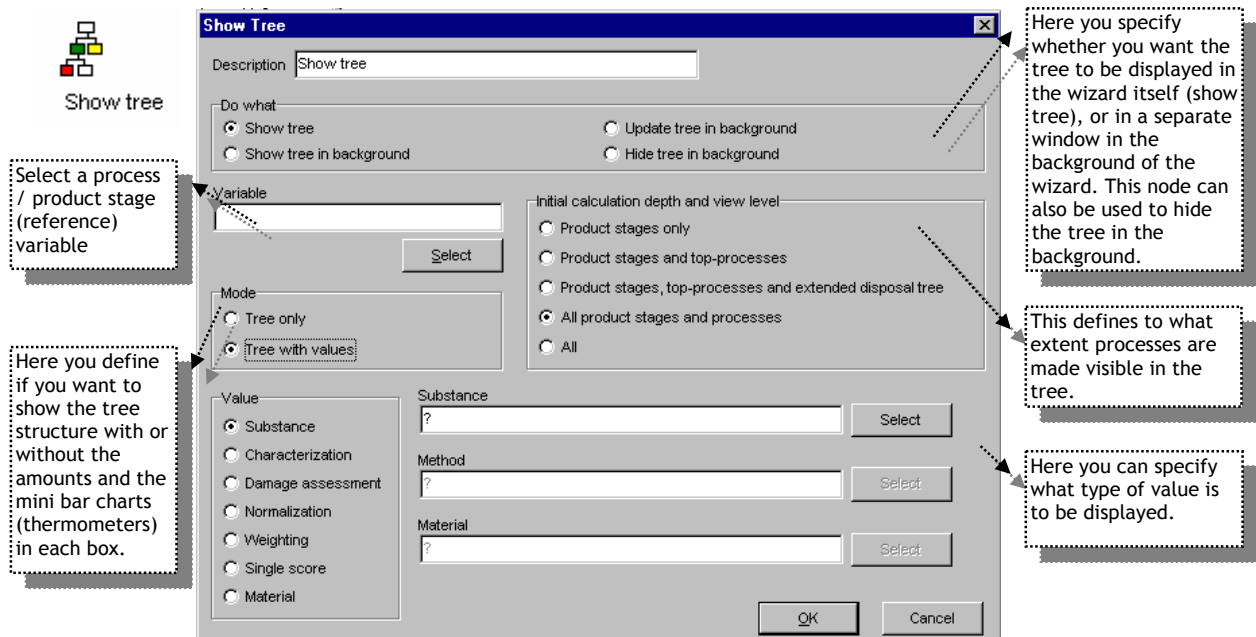


Figure 20 Window of a Show tree node

In the tree node you specify what results must be presented, and how this must be done. How refers both the definition of options available to the user, and the way the tree is displayed initially.

If you choose 'tree with values' under 'Mode', you can specify what type of value is to be displayed. Depending on your selection you can specify a specific substance, results calculated with a certain impact assessment method, or a material. If you specified the mini bar chart must display the results of an impact assessment method, you can specify the level of aggregation under 'Value'.

2.2 Variables

While the user goes through the wizards, he enters data. This data is stored in variables. In addition, the wizard nodes themselves can create new information that needs to be stored in variables.

All data collected or generated by the wizard can be stored in variables. There are 7 types of variables, each with different characteristics. These are explained in the following sections.

The variables are stored in the same library or project in which your wizards are stored. During the development of the wizards check carefully and document which variable is used for exactly which purpose. A *debugging* window is available to see the contents of the variables while testing a wizard.

2.2.1 Product stage Variable or Process Variable

This variable refers to all information in a *new* product stage or process created by the wizard. The variable itself is identified with a *Name of the product stage or process*, *Value (number of product stages)* and a *Unit*. For instance, a process can be identified as Aluminum, 12, kg. In the *operations* node, the data that is stored in the variable can be edited or extended.

2.2.2 Product stage or Process REFERENCE variables

This type of variable is very similar to the previous type with one subtle difference. It refers to existing process stages or processes that are stored in the database. This means you cannot edit this stage of process from within the wizard, but you can make a reference to it. The variable itself is identified with a *Name of the product stage or process*, *Value (number of product stages)* and a *Unit*. For instance, a process can be identified as Aluminum ETH, 12, kg. References can be used if you want the user to choose from a number of predefined processes. The selected reference is then copied (assigned) to a variable for further processing.

2.2.3 Text and numeric variable

A text variable can contain a text of maximum 30 characters. A numeric variable contains a value plus unit. Text and numeric variable can be used to store information entered by the user (in an 'enter name' node or an 'enter value' node). They can also be used in operations to store temporary results (see chapter 3 Examples and Practice).

2.2.4 Results variable

This variable contains calculation results, such as effect scores or LCI results. Results variables are created in a calculation node and are needed as input in show chart/table nodes and show LCI results node

2.2.5 Linking information to variables

A product stage (reference) variable, and a process (reference) variable can contain both a link to a process or product stage, and an amount with unit. This is useful, since SimaPro uses both types of information to build the model.

The first step is always to start by making a link or assigning a process or product stage. The second step is to link or assign an amount with unit. You cannot start with the amount first, as you will not get the right results!

2.3 Nodes and variables

2.3.1 Relation between nodes and variables

Nodes and variables are related in the sense that specific nodes, like select process or product stage, use specific variables. In Figure 21 and, you find an overview of the relation between variables, and between variables and nodes.

To start with the relation between variables and nodes, from bottom to top of Figure 21:

- Results from a *calculate node* are stored in a *results* variable
- The values a wizard user enters in an *enter values* node are stored in a *numeric* variable
- The process or product stage a wizard user selects in a *select process/product stage* node are stored in either a *process reference* variable or a *product stage reference* variable
- Processes or product stages created by the wizard in an *operation* node are stored in a *process* variable or a *product stage* variable

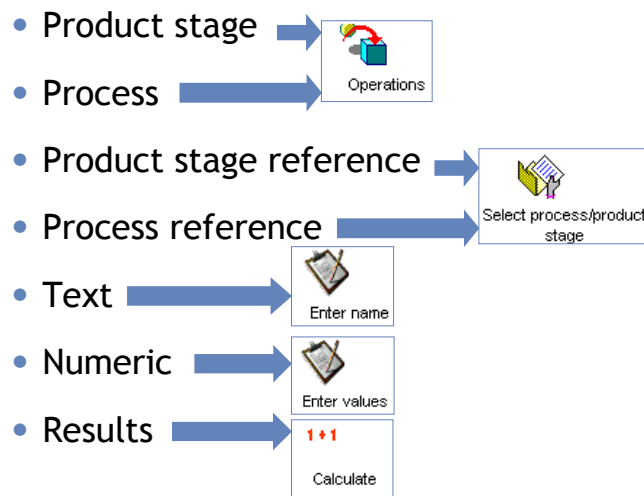


Figure 21 Relation between variables and nodes. Nodes are presented as a graph, variables in text.

In Figure 22 you see the relation between variables and analyze/ show results nodes. Process (reference) and product stage (reference) variables are the input for calculation nodes and show tree nodes. The results of a calculation node are stored in a results variable, and this variable is the input for a show chart/table node or a show LCA results node.

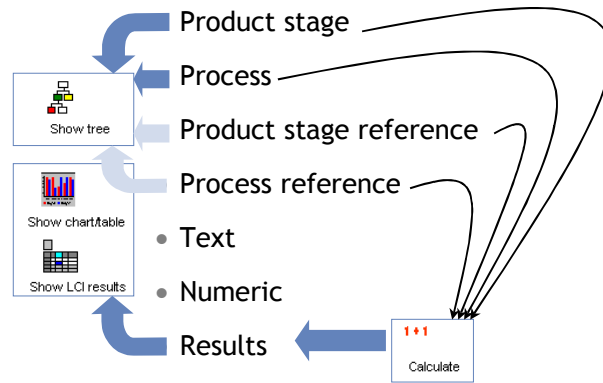


Figure 22 Relation between variables and nodes. Nodes are presented as a graph, variables in text.

Variables have different kinds of possible relationships as well (see Figure 23):

- Values stored in *numeric* variables can be linked to *process (reference)* variables and *product stage (reference)* variables [- - -]. Using this option you can attach an amount to a process or product stage.
- Text in a *text* variable can be linked to *process* variables and *product stage* variables [____]. Using this, you can give process and product stages a name, or use the text typed in by the user to name processes or product stages created in the wizard.
- *Process reference* variables can be linked to *process* variables and *product stage* variables [_ _ _]. This means you can link processes stored in the database to create new processes or product stages in the wizard. For example, you can link a certain amount (numeric variable) to electricity or transport in the database (process reference variable) and link this to a new production process (process variable).
- *Product stage reference* variables can be linked to *product stage* variables [...]. This means you can link product stages stored in the database to new product stages created in the wizard. For example you can use assemblies stored in the database as subassemblies of a new assembly created in the wizard.

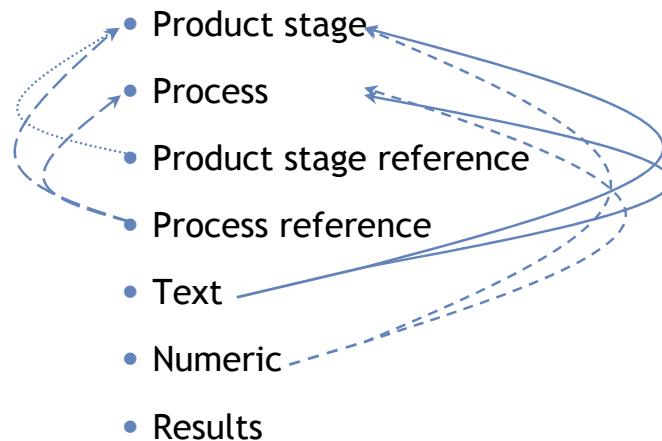


Figure 23 Relation between variables

2.3.2 Text and numeric parameters in texts

You can insert parameters in a text message to the user that can be influenced by a wizard. For example, you ask the user to specify two product names, for example, a name SIMA for a plastic and a name PRO for an aluminum coffee machine. Later in the wizard you can include a sentence, Please specify the lifetime for Model %s, in which %s is replaced by 'Sima'.

The text (string) and numeric (value) parameters are always referred to by %s (strings) and %d or (for values). They cannot have specific names.

- %s Places the text contained in a text variable in the text field where you have typed the text-code.
- %d Places the value and unit contained in a numeric variable in the text field where you have typed the text-code.

The string parameter allows the wizard writer to add variable texts to any node with a text field. During the execution of the wizard, SimaPro will replace the string parameter code by text that has been assigned to a String variable.

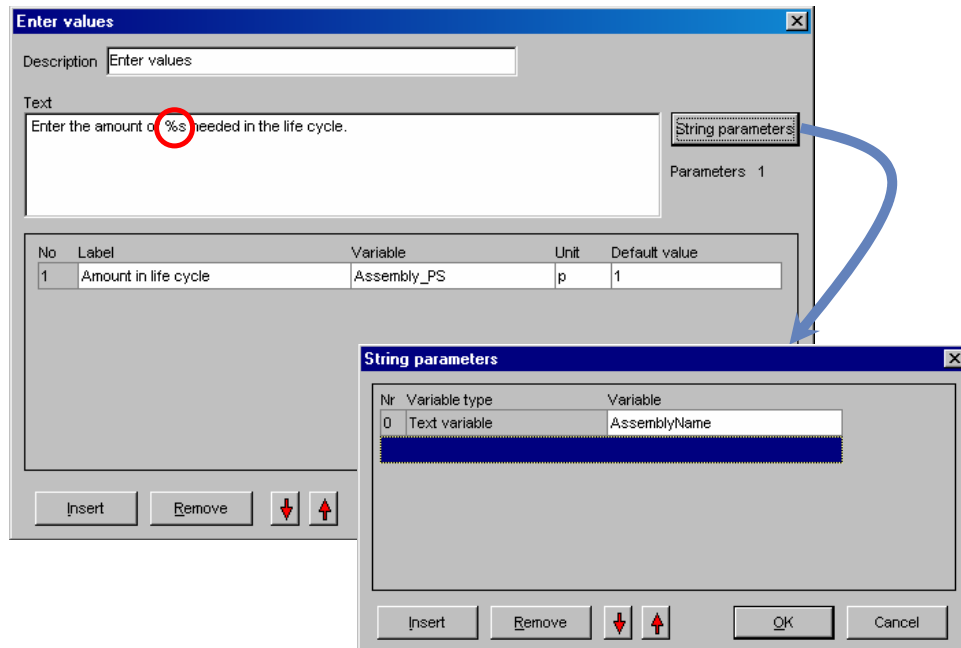


Figure 24 Example of the use of string parameters in the text field of a node

For example: You can ask the user in the enter name node to give a name of a product. This name is assigned to a text variable. Now that the name is known, you can let this name appear in any text in the following nodes. For example, you can make a text field like: How many screws are in %s?
When the wizard is run it will not display %s but the value the user has assigned to the string variable that belongs to %s.

Apart for the *text variables* there are also *numeric variables*. These are used to display numbers. For example, you have asked about the number of legs of the table in an *enter value node*. If the user has replied 4, this can be stored in a value variable. Later in the wizard you can ask the question: are all %d legs painted? When the wizard is executed the user will see that %d is replaced by the number 4.

If you want to include a variable, you simply type % s of %d within the text. You can repeat this as many times as you want within one node. Do not use any other characters; just use %s and %d as often as needed.

When you use %s or %d in the text, you have to click the sting variable button, which is available on each node with a text field. A window will appear that lists the string and value variables that are relevant for this node (see Figure 24). In this window you can select the variables whose content should be displayed in the text. Use the insert button to create or link to a variable. You will be shown a new box with an overview of all the variables currently available. Select one, or create a new one and close that window. The variable will now appear in the list. If you need more than one variable, as you have used

the %d or %s sign more than once in this node, repeat this operation. The order in which the string or value parameters are listed determines which variable is linked to which parameters.

Use the arrow buttons to place the variables in the correct order.



2.4 Programming wizards

Wizards are programmed in the ‘wizards section’ of the explorer in SimaPro (see Figure 25). The buttons on the right sight of the screen can be used to create new wizards. You can also edit or copy existing wizards.

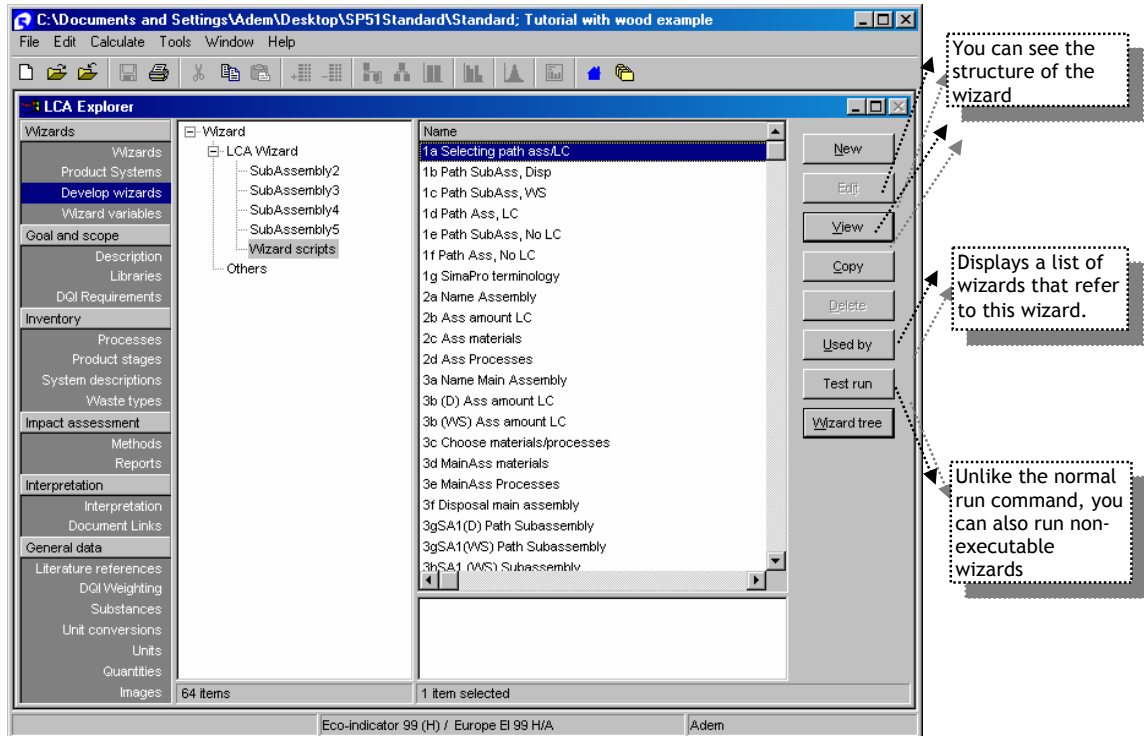


Figure 25 Window of the wizard section with an explanation of the buttons on the right side of the screen

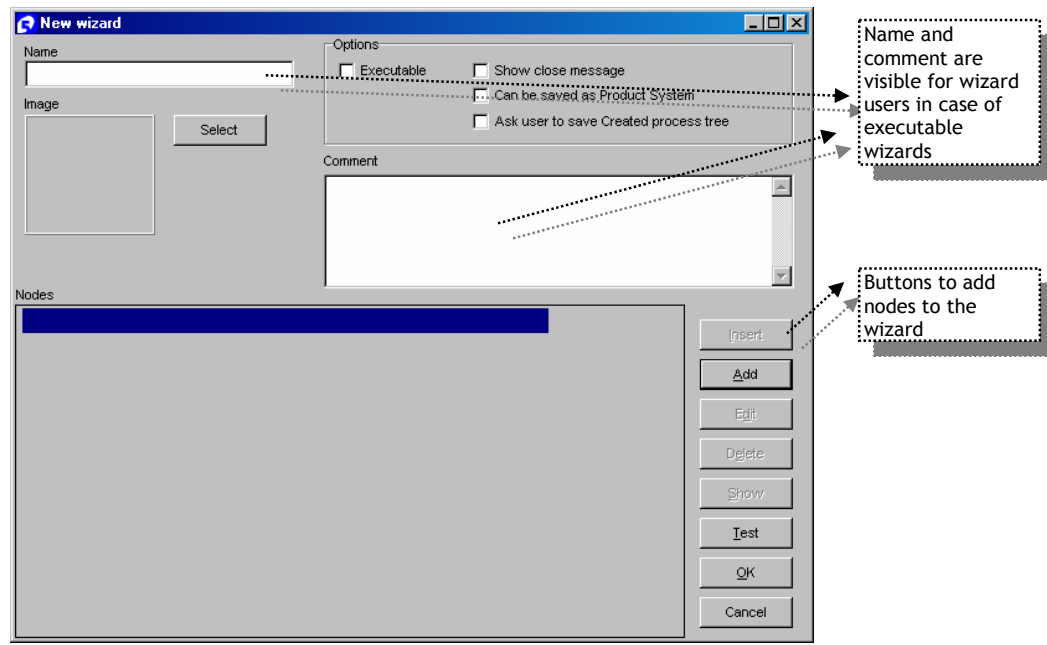


Figure 26 Partial view of the window creating a new wizard

Figure 26 shows a part of the window you will see when clicking the ‘New’ button to create a new wizard. Here you can give a name to the wizard, and write the purpose of this wizard under ‘comment’. The content of the comment field is visible for wizard users in the case of executable wizards. This is similar to the comment field of processes. For non-executable wizards, you can use the comment field for your own documentation.

The buttons on the right side of the screen can be used to add nodes.

2.4.1 Debugging

The debug tools help you to trace what is happening while you test run the wizard. The debugging windows are automatically opened if you use the test run button. Three windows open, which show data while you test run the wizard.

Node results window

This window shows all the basic steps performed by the wizard nodes. This window is particularly useful if you build up your wizard from different smaller wizards that are branched. It enables you to see which wizards are used, in which order, and which results are obtained from every step.

For example, if you run the guided tour wizard, step 5 calls another wizard. This wizard creates a number of objects. By clicking the next button to go to step 6, you will see a number of steps from the Create Objects sub wizard.

Variable values window

This window shows the content of the *variables* that are used in the wizard. It allows you to validate the correctness of all intermediate results.

When an error occurs SimaPro will generate a message stating in which node, from which sub wizard, the error occurred.

View Script Stack window

This window shows the content of the *scripts* that are used in the wizard.

2.4.2 Wizard structure

You can build wizards in a modular way, as you can jump from one wizard to another. This allows for efficient programming, as you can define a number of sub wizards that perform some tasks that can be repeated. This jumping also allows you to design alternative routes or branches through the wizard. For example, you can ask the user in a certain node if he wants to compare two products, or to define another product. Depending on the answer, another sub wizard is activated as the next node.

To jump from one wizard to another, you use the *call wizard* node. See for example Figure 27. The wizard is started on the left side. A *call wizard* node activates wizard A. Once all nodes from wizard A are done, the wizard jumps back to the horizontal branch and follows the nodes in that first branch.

A wizard can have similar parts. You can make a copy of a wizard, and use call wizard to connect all 'sub wizards'. For example you design a wizard in which the wizard user can model two different production processes, in order to make a comparison in the end. In the wizard, you first want to collect data from one production process, and later collect data from the alternative production process. The nodes required to enter the data will be similar for both production processes, however the guidance text might be slightly different. In that case you can model a wizard for the first production process, and make a copy of that wizard in which you make some changes to create the wizard for the alternative production process. This situation is shown in Figure 27.

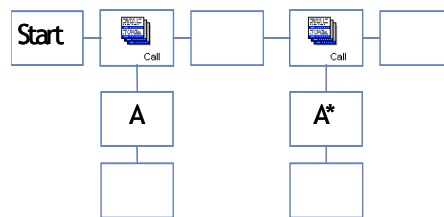


Figure 27 Schematic overview of the structure of a wizard with similar branches (A and A*)

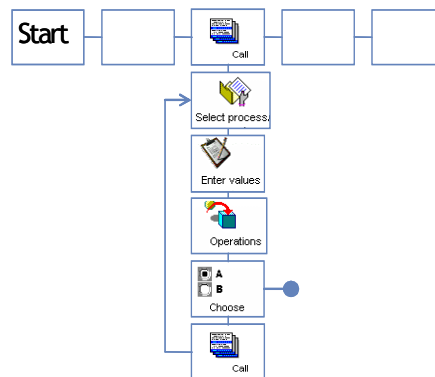


Figure 28 Schematic overview of a wizard with a repetitive branche.

You can also use the call wizard node to repeat a part of the wizard. For example you may want to enable the wizard user to add as many materials to a new assembly as he or she would like to do. For this part of your wizard, you program a branch of the wizard in which the user can select a material process and enter the amount of material in the assembly. In an operation node you link the process with the appropriate amount to the assembly. Then you can ask the user in a choose node whether he or she would like to add more materials to the assembly. If the user wants to add more materials, a call wizard follows, in which the wizard will call itself and the wizard will start all over. If the user chooses not to do so, the wizard ends. If this branch wizard is used in another wizard, it will jump back to the next node after the *call 'add material' wizard* node. This situation is shown schematically in Figure 28.

The combination of *choose-* and *call wizard* nodes can also be used to create a flexible wizard structure. Suppose you model an assembly in a wizard and before analyzing, you want to give the user the option whether a life cycle with or without a waste scenario should be modeled as well.

Figure 29 gives you some options to model a flexible structure in which the user can choose what will be modeled. In this example the user models an assembly and has the option to model a life cycle and a waste scenario. On the left side, the user chooses directly after modeling the assembly what he or she will do. Depending on the choice, the wizard will call other sub wizards. A choose node is always the end of the 'main' wizard, therefore the 'analyze' wizard has to be called in the three branches of the choose node.

To prevent ending of the 'main branch' by a choose node, a structure such as that shown in Figure 29 on the right side can be modeled. Before choose, a call scrip is introduced.

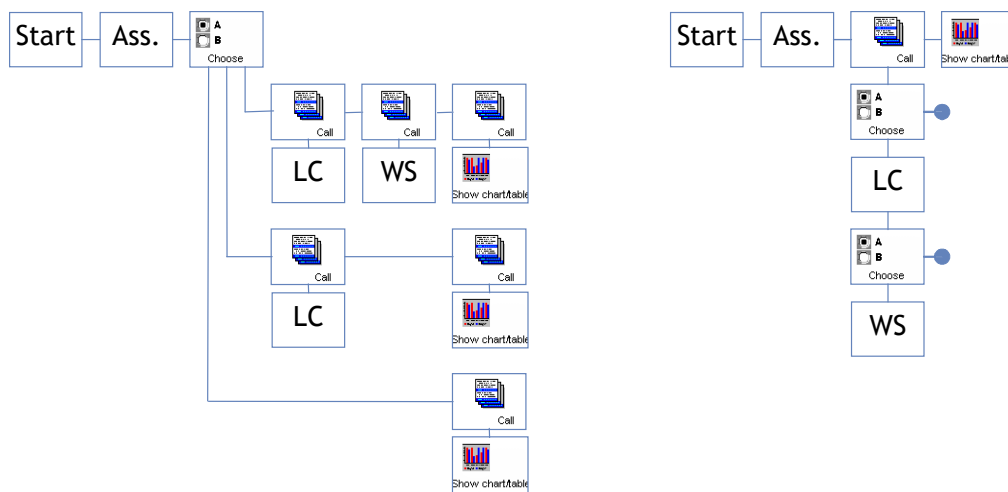


Figure 29 Flexible wizard structures in which the user has options to choose what wizard route should be followed. Example where the wizard user models an assembly (Ass.) and can choose to model a life cycle (LC) and a waste scenario (WS).

2.4.3 Other things you should know....

Limitations in the use of nodes

- You cannot copy or move nodes. However, you can copy wizards. Prevent 'long' wizards, use 'call wizard' to connect short (sub) wizards
- If you delete a choice, you delete all following nodes. Be careful in creating a long wizard after a choose node
- 'Choose nodes' can only be inserted at the end of a wizard
- Nodes are not saved until you save the wizard

Wizard design

Changing a wizard that is already built is a hard task. Therefore before programming a wizard, it is useful to start with a 'goal and scope' of the wizard you are making. In this goal and scope you can deal with the following aspects:

- What is the purpose of the wizard (demonstration, easy analysis of new products, etc.)?
- What is the audience and level of information input (see also section 1.3.1)
- Which parameters are flexible for the user
- What does the wizard create
- What are the possible routes through the wizard
- What results are presented in the wizard

It is helpful if you perform an LCA of the wizard topic first, to know the crucial aspects of the life cycle.

If you make a wizard for a client, you can discuss the goal and scope with your client before you start building it. This has proven to be useful, since the client will get an idea of the result of the wizard at the beginning of the project, and can help in designing a wizard that fulfils the expectations.

After designing the wizard on paper, build the technical node structure first. Check whether the wizard is working and gives the expected results. Add messages and text later on. If you have similar branches in your wizard, build one first and copy when finished.

Storing data in the database

Processes and project stages created in the wizard can be stored in the SimaPro database. Items created by the wizard are stored in the project that is opened when the user starts the wizard.

Processes are stored while running the wizard; product stages are stored after ending the wizard. Thus, if the wizard user cancels the wizard while running, product stages are not saved!

3 Examples and Practice

3.1 Example: Assembly

The example of practical use of variables and nodes to build a wizard is explained based on the LCA wizard in SimaPro. In this example we create a small wizard in which an assembly is created. The wizard user should enter the name of the assembly.

To do so, we will need two variables:

- A variable to store the name of the assembly. This will be a text variable, and in this example we will call it 'AssemblyName'
- A variable to store the product stage assembly. This is a product stage variable, and we choose to call it 'Assembly_PS'

We start our wizard with an enter name node. In Figure 30 you can read what we did step-by-step.

Figure 30 Enter name node in a wizard example

The text within the red lines in Figure 30 will be visible for the wizard user, as you can see in Figure 31.

Figure 31 The 'enter name node' of

Figure 30, as seen by the wizard user

The next step in the wizard is to create an assembly with the name given by the wizard user. We add an operation node. Under destination we put the variable for the product stage. By doing this, a product stage is created by the wizard. We double click under Source 1 to select the text variable with the name of the assembly. Under operation, we select assign, to give the assembly the name stored in 'AssemblyName'.

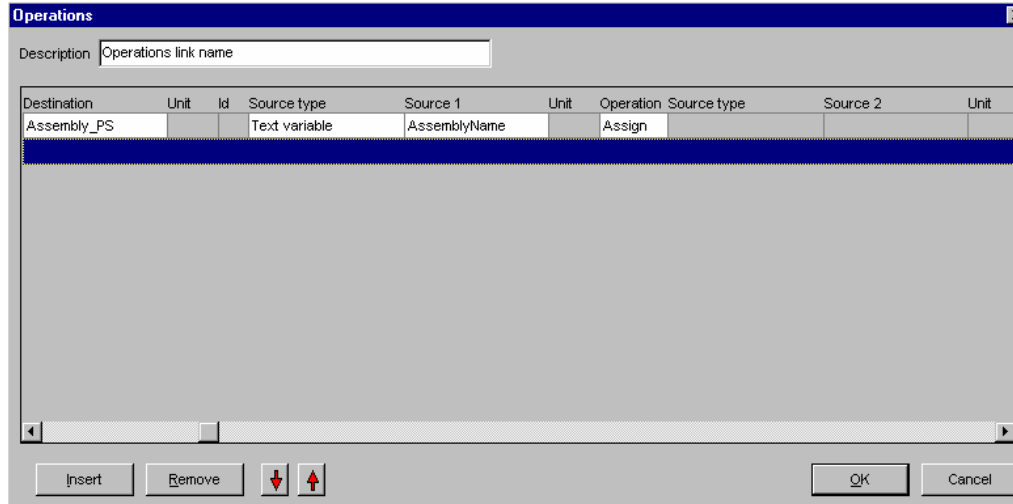


Figure 32 Operations node to create an assembly product stage with the name given by the wizard user (stored in variable AssemblyName)

The wizard should continue by asking the user how many assemblies are needed in the life cycle. Then a life cycle should be created that contains the right amount of assemblies.

To do so, we will need some new variables:

- A product stage variable to store the life cycle product stage: LifeCycle_PS
- A text variable for the name of the life cycle: LifeCycleName

We will link the number of assemblies needed in the life cycle directly to the product stage variable 'Assembly_PS', as this variable can contain an amount beside other types of information. This will be done in an *enter values* node (see Figure 35)

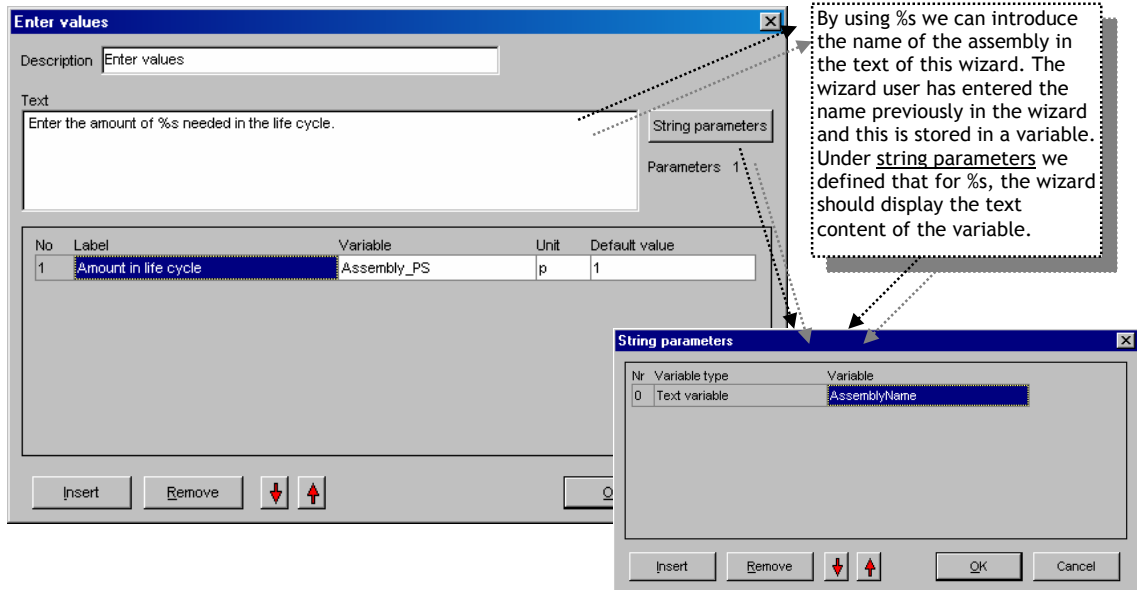


Figure 33 Example of an enter values node where the user can enter the amount of assemblies needed in a life cycle

Suppose the wizard user has named the assembly 'table'. In that case the 'enter name node' of Figure 33 will appear to the wizard user as shown in Figure 34.

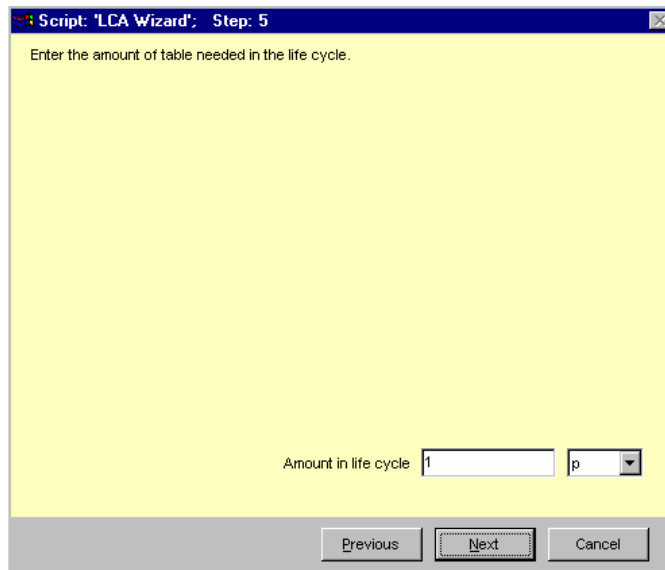


Figure 34 Enter name node of Figure 33 as it will be shown to the wizard user (The wizard user named the assembly 'table')

Now that we know the amount of assemblies needed in the life cycle, we can create the life cycle and link the assembly to the life cycle. This can be done in an operations node.

First we create the name of the life cycle by putting the text 'Life cycle_' before the name of the assembly. This life cycle name is stored in the text variable LifeCycleName. In the second line of Figure 35 we assign that name to the life cycle product stage. By doing so, the wizard will create this product stage. In the third line we link the assembly

to the life cycle. Since the amount is stored in the variable 'Assembly_PS' as well, the wizard will link the appropriate amount of assemblies.

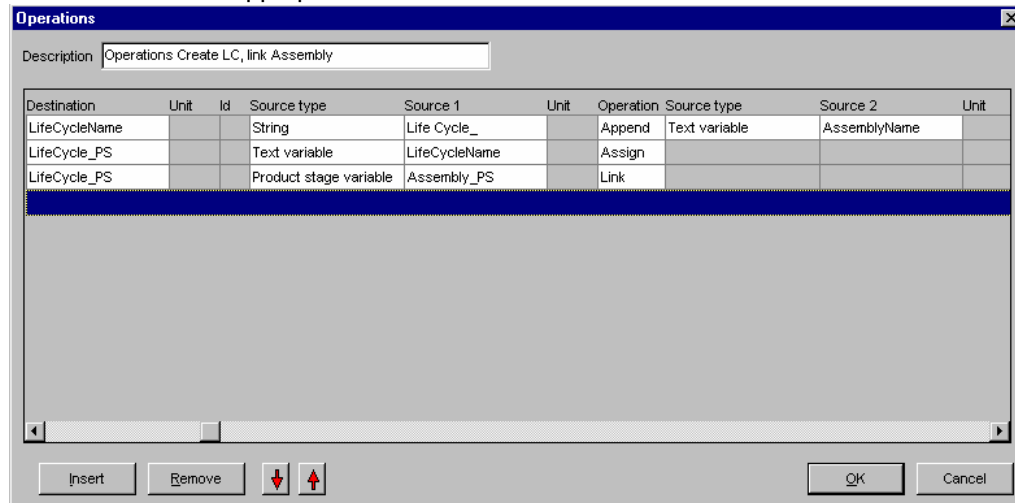


Figure 35 Operation to create a life cycle and link an assembly

3.2 Example: calculations

Operations can be used to perform calculations in the wizard. Because of the fact that the wizard can perform calculations, wizard users can be asked to enter information on a level that is logical to them. The wizard to calculate the amounts that are meaningful for SimaPro can use these data.

Consider the example of a wooden plank. To perform a life cycle assessment, SimaPro needs input on the weight of the wood. Suppose the user has access to the measures of the wood, like length, thickness and width. We ask the user to enter these values and store them in three variables: 'WoodLength', 'WoodWidth' and 'WoodThick'. The wizard can use these values to calculate the weight of the plank based on the density of wood (assume 600 kg/m³).

See Figure 36 to see how such an operation is programmed. Each line can only contain one calculation step. Since we need several calculation steps, we need several lines in the operation. The results of each step are stored in a temporary numeric variable 'TempValue'. The first step is to multiply the length with the width. The second step is to multiply the result of the first line with the thickness of the wood, which results in the calculated volume of the wood. The last step is to multiply the volume with the density. Please note that the unit of the calculated result under density must be given as well.

Wizards can use the unit conversions and the factors between units of the same quantity. This means you can use different units of the same quantity in an operation node, since the wizard 'knows' that 1000 grams equal 1 kilogram.

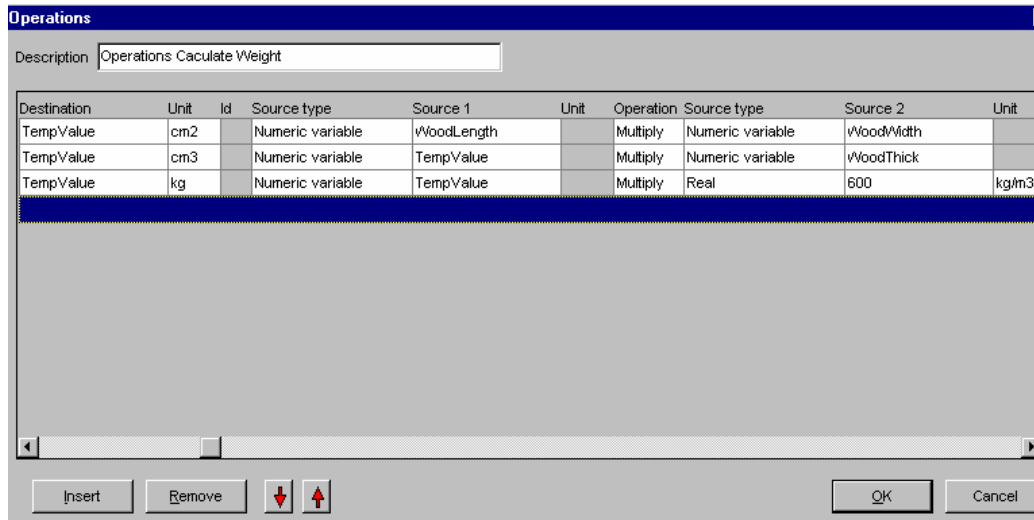


Figure 36 Operation to calculate the weight of a material based on its density and measures (length, width and thickness)

3.3 Bottle practice

You can practice wizard making yourself. Like performing LCAs in SimaPro, wizard writing is also something you learn by doing. In this section, you find an assignment to create a wizard for a glass bottle with a cap, in which the results are shown to the user.

3.3.1 Wizard programming assignments

The assignment is split into two parts: A and B. A possible solution to model such a wizard is given directly after each assignment.

Assignment A

Imagine the wizard user is a packaging designer who designs glass bottles for beer, who wants to know a little bit about the environmental impact of the glass container.

In the wizard, you will start asking the wizard user to enter the weight of the bottle and the steel cap. Then you create an assembly with the name 'bottle' and link processes from libraries in your database with the appropriate amounts

Tip: Start with an operations node to link the library processes to variables. Processes must be linked to variables before amounts are linked!

For assignment A, you can make a wizard that looks like the wizard displayed in Figure 37. This wizard has three nodes, that are shown in Figure 38 to Figure 40.

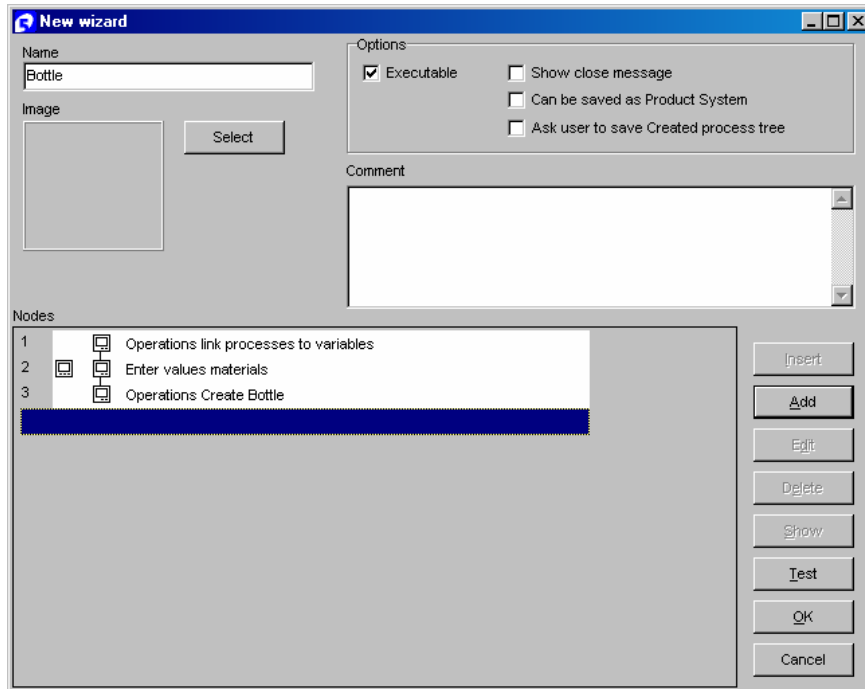


Figure 37 Wizard to model a bottle with three nodes

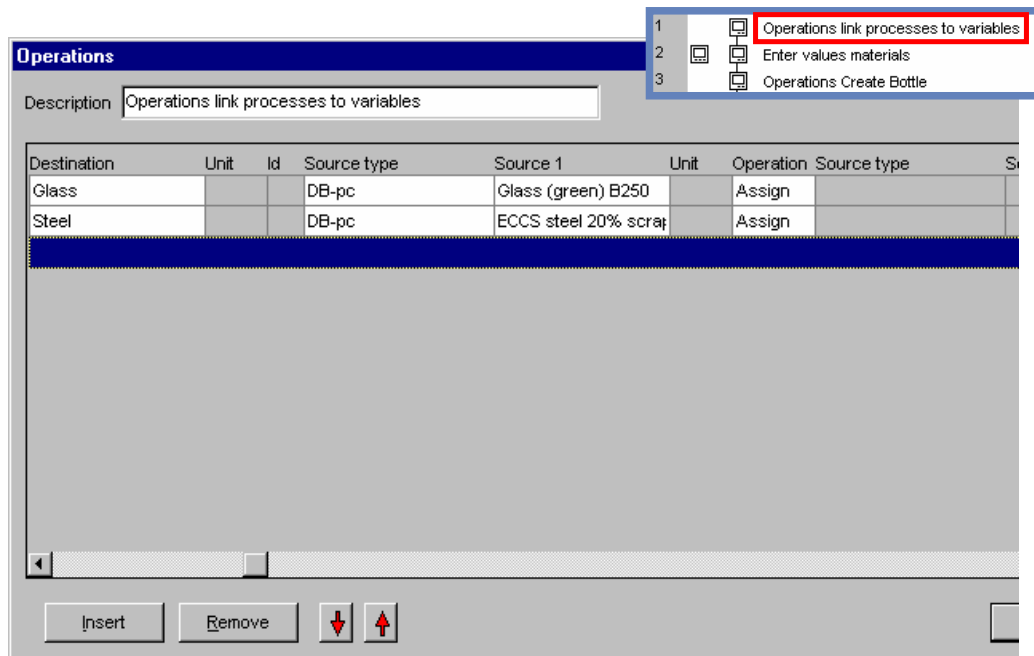


Figure 38 Step 1 to model a bottle wizard: attach processes from the database to variables

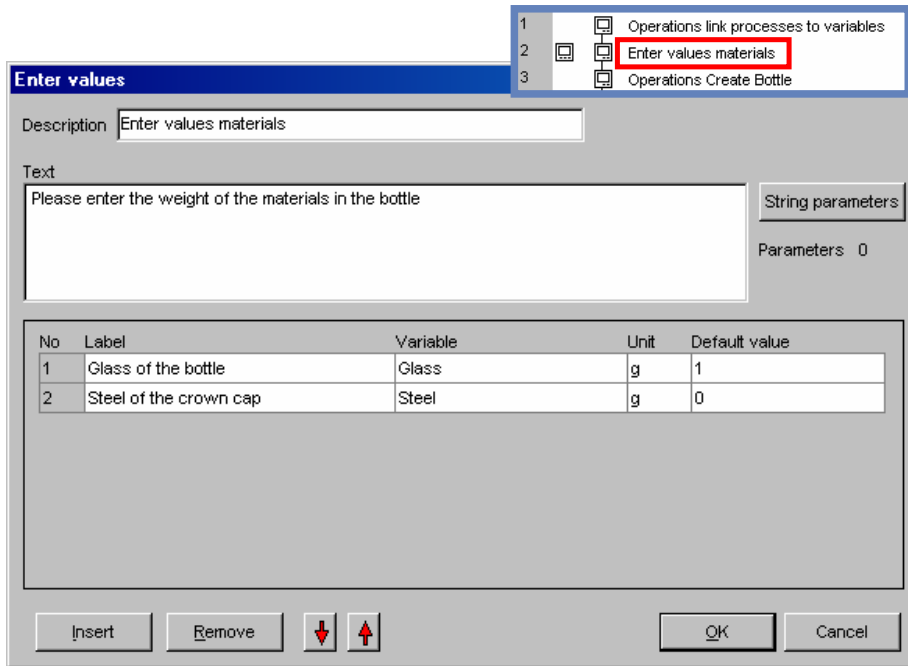


Figure 39 Step 2 to model a bottle wizard: ask to enter the weight of the materials

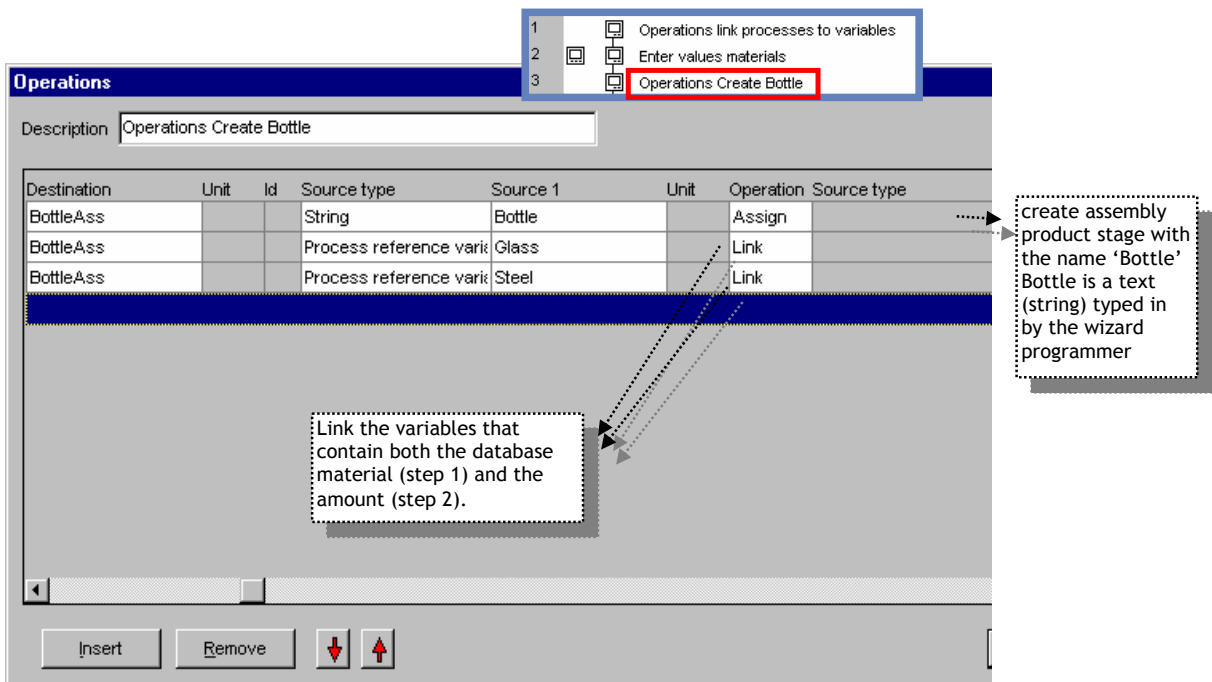


Figure 40 Step 3 to model a bottle wizard: operation to create product stage and link materials

Assignment B

In the next step of the wizard, you want to show the designer the environmental impact of the bottle. Therefore you make an analysis with the impact assessment method Eco-indicator 99 (H/A).

Before ending the wizard, you delete the bottle assembly, to prevent errors. An assembly with the same name (bottle) should not be stored twice in the SimaPro database.

To make assignment B, the wizard is expanded with three more nodes. These are shown in

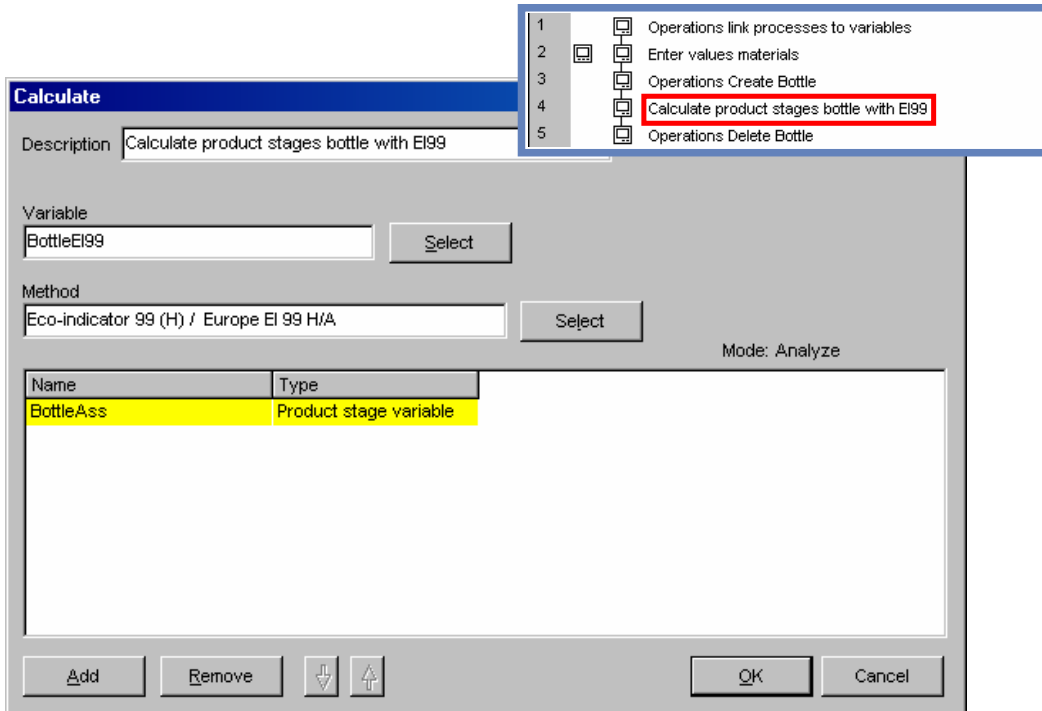


Figure 41 to Figure 43.

Figure 41 Step 4 to model a bottle wizard: Calculate the results of an analysis with Eco-indicator 99 and store the results in a variable

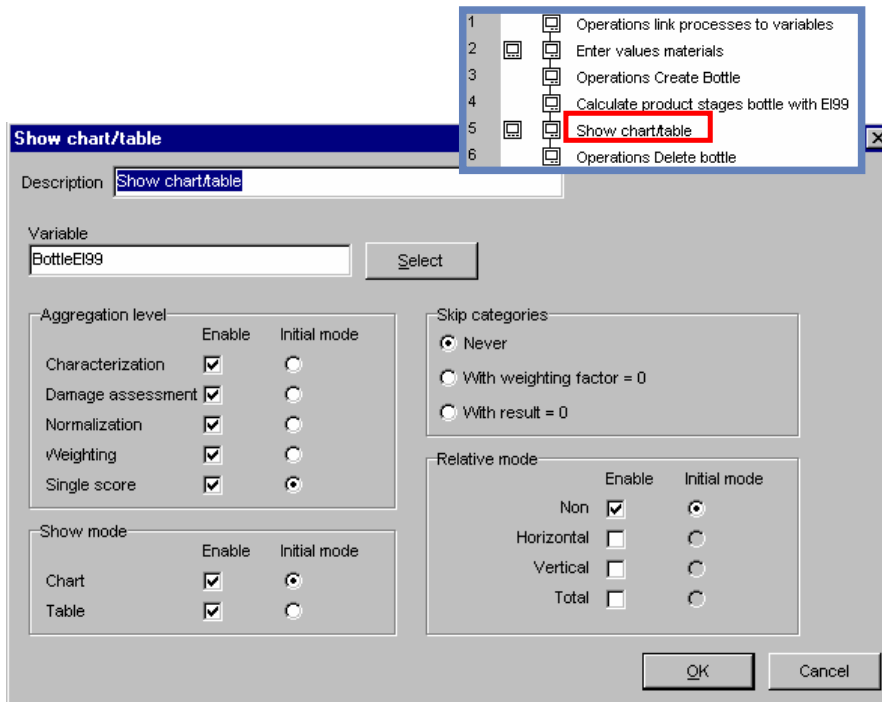


Figure 42 Step 5 to model a bottle wizard: show the calculated results stored in variable 'BottleEI99' in a chart to the wizard user

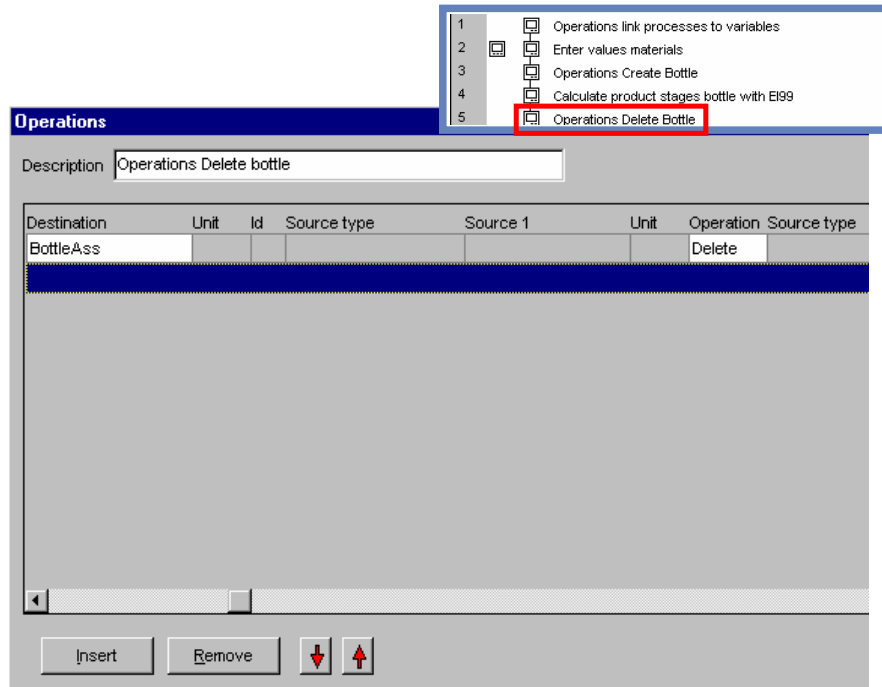


Figure 43 Step 6 to model a bottle wizard: delete the product stage created by the wizard

3.3.2 The result: what the user sees

As a result of assignment A and B, the user will see three wizard windows. In the first one the user can enter the weight of the materials, the second one displays the results. The last window is automatically generated and reports that the wizard is finished.

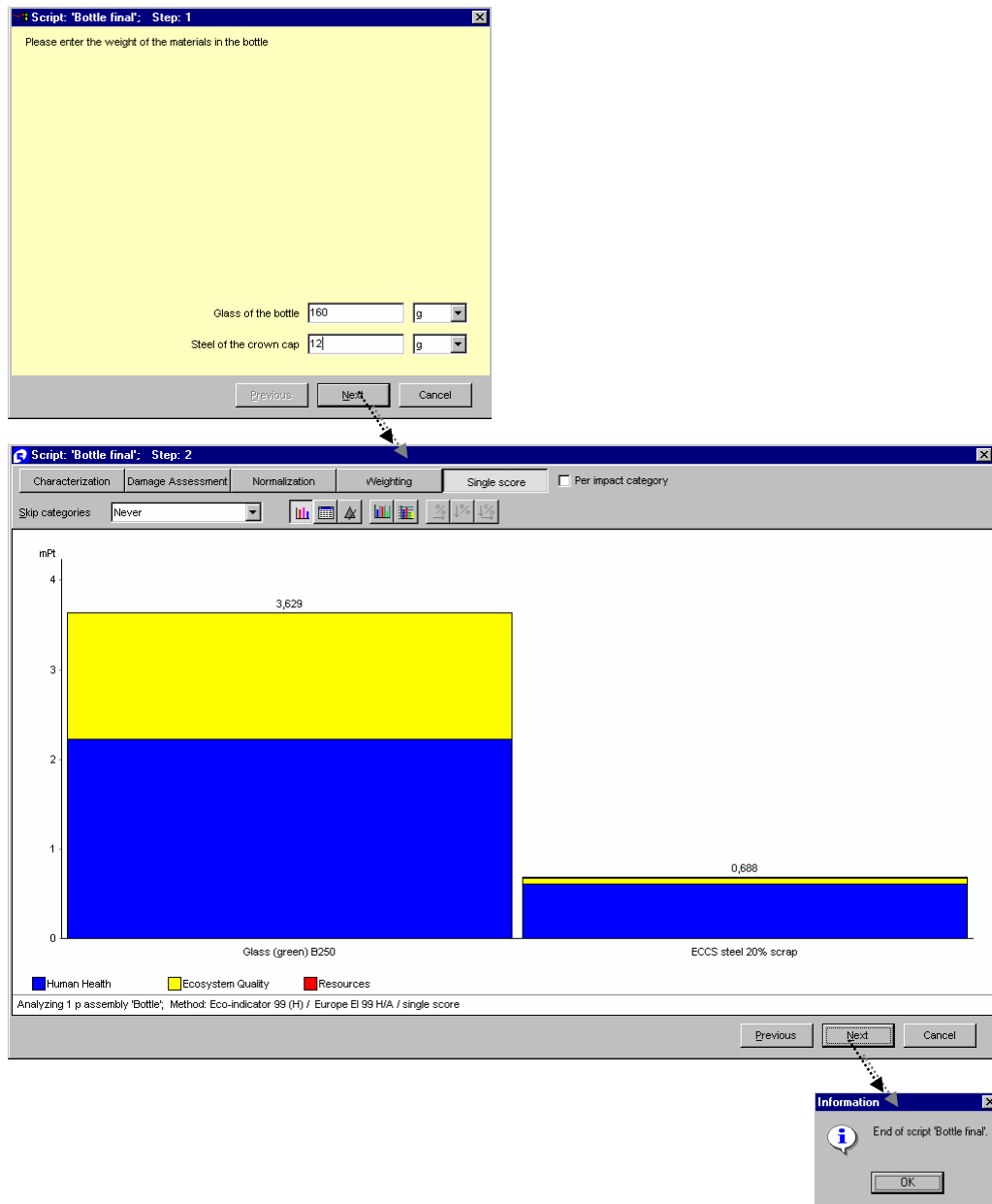


Figure 44 Windows of the bottle wizard as they are presented to the wizard user

4 Product Systems

The answers of a wizard can be saved in a 'product system' in the category 'Wizard' in the SimaPro explorer. A product system presents on one page all of the answers a wizard user gave while running the wizard. The user can 'run' a product system and make changes to the values entered in the wizard. These changes can be saved in the project he or she is working in, without having to run the wizard again. This can be a powerful feature for wizard users if they want to make small changes in a model created during previous running of a wizard.

Product systems can be created at the end of an executable wizard that is run using 'tools' in the menu bar. At the end of the wizard the user sees a window in which he or she can specify the name of the product system in which to save the answers from the wizard (Figure 45).

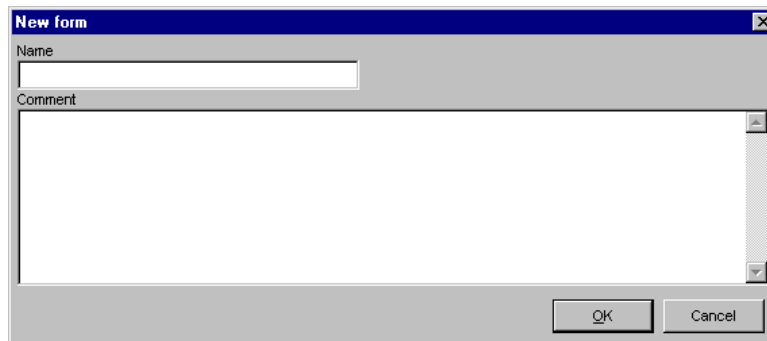


Figure 45 Window at the end of a wizard in which the wizard user can save the answers from the wizard in a product system

Product systems are stored under 'Product systems' in the wizard section of the explorer of SimaPro. Users can use the product system using the 'run' button. An example of a product system is shown in Figure 46. The user can change the data in the white field. By clicking on the show tree button, the user will see the tree of the modeled life cycle above the button. To decrease the length of a product system, the user has the option to hide the information text (see Figure 47).

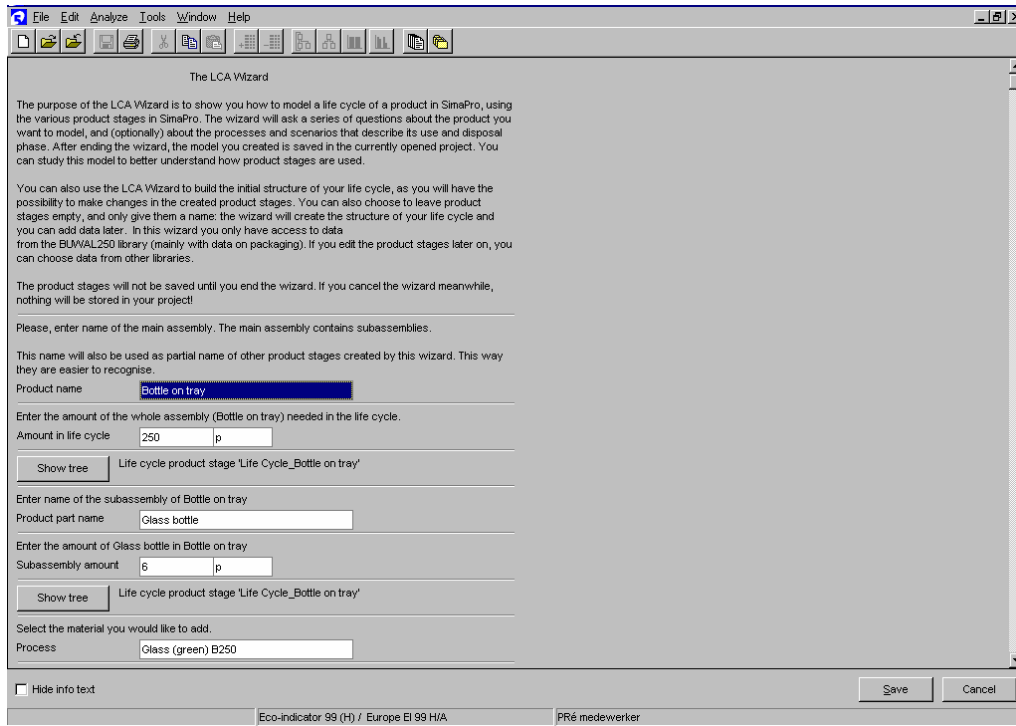


Figure 46 Window of a product system saved after running the LCA wizard

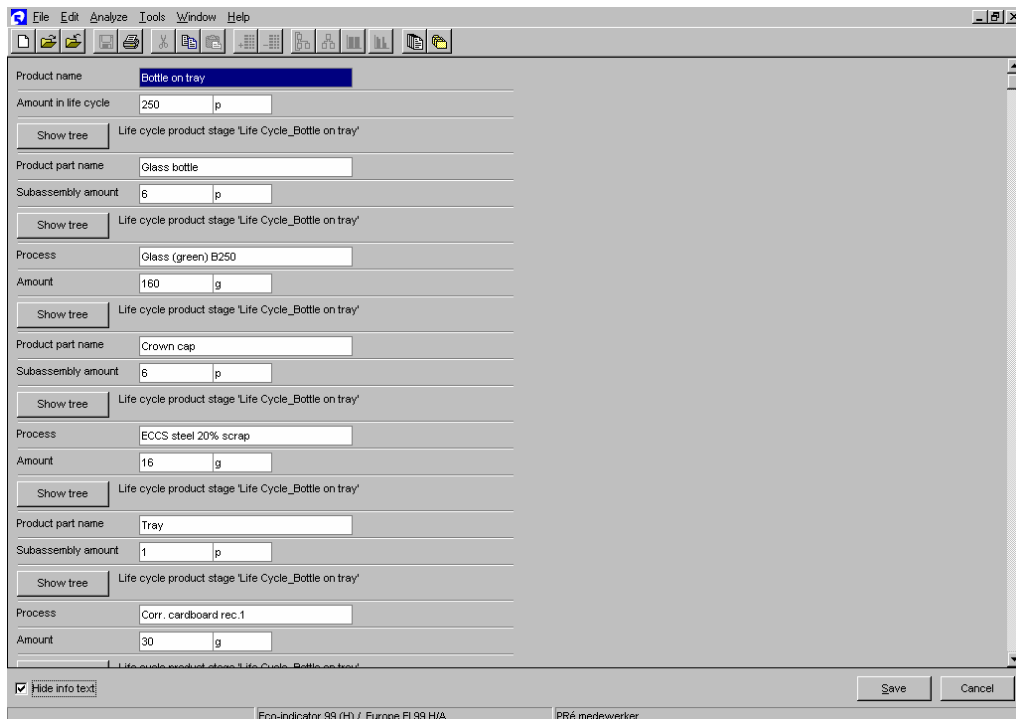


Figure 47 Window of a product system saved after running the LCA wizard, with text hidden



product ecology
consultants

PRé Consultants bv
Plotterweg 12
3821 BB Amersfoort
the Netherlands

phone + 31 33 4555022
fax + 31 33 4555024
e-mail support@pre.nl
web site www.pre.nl