

---

## LccWare V3 Technical Specification

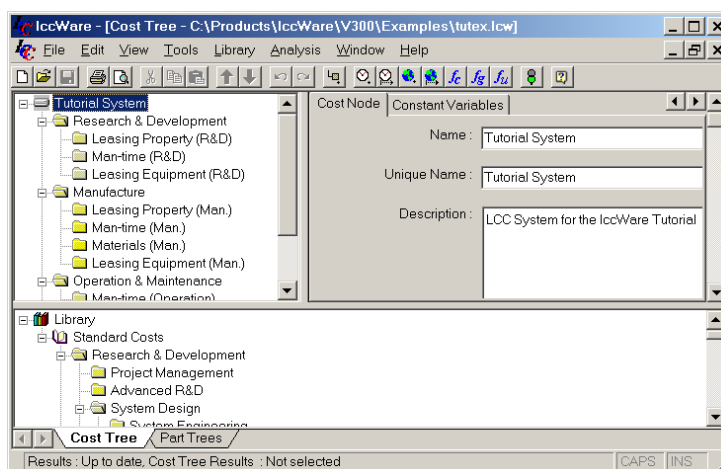
---

### Platform

Runs under Windows 95, 98, NT, 2000, Me. Recommended host memory requirements 32Mb+

### Summary of functionality :

- Life cycle costs are represented by a cost breakdown structure
- The cost breakdown structure is composed of cost nodes (cost elements) in a cost tree structure
- Cost trees are constructed interactively
- The copy, paste and drag and drop facilities allow nodes of the cost tree to be copied or moved
- Cost functions calculate the cost to be applied at each cost node in the cost tree using either global variables or variables local to the cost node as inputs
- The cost at a cost node is the cost generated by the cost function plus the cost of child nodes
- Cost functions are written in a Visual Basic compatible language
- The analysis module calculates cost and cumulative cost through time at each cost node in the cost tree for each time sequence
- Any number of time sequences and any number of time points per time sequence may be created
- The analysis module debug facility allows the user to insert breakpoints, step through cost function code and examine variable values
- Global and local variables can be constant or time dependent
- The sensitivity study option allows users to examine effect of variations in global and local variables
- A browse facility allows users to determine where the functions, nodes and variables comprising the study are referenced
- Part trees provide the user with a convenient method of assigning the costs associated with a frequently used piece of equipment to a cost function
- LccWare projects may (optionally) be protected by 4 levels of user security - administrative, design, modify or read.
- A project wizard assists the user in creating new projects
- Frequently used cost elements may be saved in a library
- Comprehensive reports interfacing with Microsoft Office products
- Extensive import and export facilities

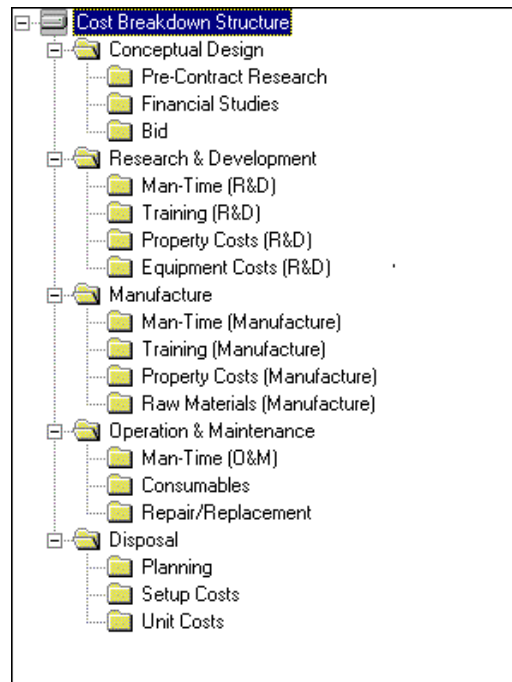


*lccWare Main Window*

## Features Overview

### LccWare

LccWare allows you to calculate the life cycle cost of a system. The total cost will be made up of several cost categories that you can define, such as Research & Development, Operation & Maintenance and Disposal. The categories can be further sub-divided to a level at which you can calculate detailed cost values. This division of the total cost into sub-categories is known as the Cost Breakdown Structure (CBS). The CBS is represented in lccWare in the form of a tree.



The nodes on the tree represent the cost elements in the CBS. LccWare allows you to build the tree interactively and create your own CBS.

LccWare allows you to create cost functions that will be used to calculate the value of a cost category. These cost functions can range from simple equations to more complex calculations based on Visual Basic compatible coding.

You can easily assign the cost functions to the nodes on the cost tree so that lccWare can calculate the individual cost values.

The costs are calculated at user defined time points within the life cycle of the system. Each set of time points is called a time sequence and has a unique name. LccWare allows you to define more than one set of time points so that you can investigate the effects of variations in the time point dates and the values of variables at those times.

You can define the variables that you wish to use in the cost calculations and supply values that are either constant over time or vary with time. You can assign values to the time varying variables at each of the time points in each time sequence. Variations may be applied to each of the variables in turn to determine the sensitivity of the study to each variable.

In addition to constant or tabular variables you can create 'global functions'. For example, these functions could be used to calculate values to be used in cost function calculations where a variable can be represented by a function.

LccWare also has the facility to create 'user functions'. These functions can be assigned to any cost node and can be used to perform ancillary calculations such as the Present Value of a cost. Any number of user functions can be assigned to a single cost node.

When you have instructed lccWare to run a cost calculation you can examine the results in standard text or graph reports. You can also prepare your own reports and save the formats for future use.

LccWare includes facilities for importing data from other software such as Microsoft Access, Microsoft Excel and text files. The software can also export to any of these file formats.

By default lccWare applies no security restrictions to the project databases it creates, however an optional security mechanism may be activated. When this mechanism is activated all project databases have security information appended. This information determines what operations individual users can perform on the lccWare project. Users are categorised as having either administrator, design, modify or read access.

### **Time Sequences**

A time sequence is the name of a set of time points. LccWare calculates the costs associated with cost nodes on the Cost Breakdown Structure at each time point in a time sequence. Time dependent variables can have different values at each time point in each time sequence, constant variables have the same value at each time point.

### **Time Points**

Time points are the dates at which cost calculations are to be carried out. A set of time points from the beginning to end of a life cycle is known as a time sequence. LccWare calculates the costs associated with cost nodes on the Cost Breakdown Structure at each time point in a time sequence. Time dependent variables can have different values at each time point in each time sequence; constant variables have the same value at each time point.

### **Global Variables**

A global variable is a variable that has a value at each time point in a time sequence and which can be used in any cost function, global function or user function in the lccWare model.

### **Cost Nodes**

Cost nodes represent the cost elements in the cost breakdown structure (CBS). The cost nodes are arranged in a tree structure, with the root node at the top of the tree. Each of the cost nodes has an associated cost function, which is selected by the user from the cost functions available in the project.

When the life cycle cost analysis is performed the cost from the cost function is assigned to the associated cost node, as well as the sum of the costs from the child cost nodes. The analysis proceeds from the bottom of the tree to the top, with the root cost node representing the sum of all the cost node costs.

### **Part Trees**

Part trees provide the user with a convenient method of assigning the costs associated with a frequently used piece of equipment (and its constituent parts) to a cost function.

The part trees window allows the user to add as many part trees as required at the top level. Each part tree has any number of constant or time dependent variables associated with it (known as part tree variables).

Below each part tree the user may create a tree structure of parts, as complex or simple as required. Each part has a number of values associated with it, each corresponding to a parent part tree variable (time dependent variable values are dimensioned according to the number of time sequences and the number of time points per time sequence). The user enters data values for each part/variable combination.

The values from the part trees are included in the cost calculation using a predefined function, PartTreeValue (Part Tree Name, Part Name, Variable Name, Mode). A typical cost function using a part tree value would be:

$$\text{EngineSparesCost} = \text{PartTreeValue}(\text{"Car"}, \text{"Engine"}, \text{"SparesCost"}, \text{"SUM"})$$

This is based on a part tree called Car with sub-systems including the Engine (which may itself have many sub-systems). One of the variables is SparesCost. This function call will sum all the values of

SparesCost below and including Engine. If the “SUM” argument is changed to “PART” then only the value of SparesCost will be generated.

### **Cost Functions**

A cost function is an equation or Visual Basic compatible function that is used to calculate the cost of a particular node in a cost tree. A cost function can contain global functions, predefined variables, cost node references, local variables, global variables and references to part tree variables. A cost function is evaluated at each time point in a time sequence. Time dependent variables can have different values at each time point in each time sequence. Constant variables have the same value at each time point.

### **Local Variables**

A local variable can only be used with the cost function in which it was originally defined. The cost function can be assigned to any cost node and different values of the local variables could be assigned for each of these instances.

### **Cost Node References**

A cost node reference is the name given to a child cost node, which is then used to represent the cost of the child node in a cost function assigned to the parent cost node. In normal circumstances the cost of a parent node is the sum of the costs of the child nodes. However, lccWare gives you the flexibility to override this in the cost function assigned to the parent node.

### **Global Functions**

Global functions are Visual Basic like functions that can be used to carry out certain operations or calculate values. Global functions can be used in cost or user functions. A set of pre-named global functions is supplied with lccWare that are automatically executed at certain stages of the calculation.

### **User Functions**

User functions are Visual Basic like functions that can be used to carry out certain operations or calculate values. User functions are assigned to cost nodes. An example might be to calculate the ‘Present Value’ of the cost at the cost node.

### **Library Facility**

#### *Using the Library*

The library facility allows the user to create many cost structures in a single master library. The library facility is enhanced by the cost category feature: in addition to adding cost nodes to a cost tree the user may also add cost categories. Cost categories are either children of the root node or children of another cost category. The cost category feature allows the user to organise their library in a structured and logical manner.

In addition to creating cost node structures in the library the user may also create part tree structures.

Isograph provide a library already containing the commonly used cost elements. The library may be extended by the user adding their own cost categories, cost nodes, and part trees.

The library window forms part of the main lccWare window. The user may use copy and paste or drag and drop to transfer cost nodes, part trees or parts from the library window to the cost tree or part trees window. All associated information is copied with the selected cost nodes (associated cost and user functions, variables, part trees etcetera).

#### *Modifying the Library*

The library is modified using the *Library – Modify* menu option. Selecting this option closes the current project and displays the library in the standard cost tree and part tree windows. The library facility allows the user access to all the standard lccWare facilities, except that time sequences and time points can not be added to the library. A single fixed time sequence and set of time points are automatically provided for testing purposes.

When modifying a library any existing project can be opened and displayed in the library window,

which is where the library is normally displayed when modifying projects. This feature allows the user to copy cost structures from existing projects to the library.

After modifying the library it is recommended that it is tested using the *Consistency Check* and *Run Analysis* dialogs.

### **Project Wizard**

The project wizard simplifies the creation of new projects. The wizard is based on allowing the user to select cost nodes from the master library and creating the new project based on these cost nodes.

The wizard prompts the user for the following information:

Project options – date format, system description, compiled by and approved by.

Number of time sequences and names.

Start date, number of time interval, and the time interval (month or year).

Which cost nodes to copy from the library.

### **Cost Calculation**

At each time point in each selected time sequence the cost of each node on the cost tree is calculated from the cost function that is assigned to that node.

### **Debugging the Analysis**

Most users will not normally use the debug facility; this would be used in complex cost models to trace the progress of the calculation through the coding. The debug process is controlled using the options on the *Debug* menu or the equivalent toolbar button. The debug process is described with reference to the *Debug* menu options but the description applies equally to the toolbar buttons.

The debug process is started by selecting the *Debug, Start Debugger* menu. The debugger will now be in break mode, with no statements having been executed. In this state breakpoints may be set in the code window using the *Debug, Toggle Breakpoint* menu option.

Execution is started by selecting the *Debug, Go* menu option. Execution continues until a breakpoint is reached or the *Debug, Pause* menu option is selected. A breakpoint may be set whilst the analysis is executing. The debugger is now in break mode and the following debug menu options may be used:

- *Toggle Breakpoint*
- *Step Into* a function, executing a single statement at a time
- *Step Over* a function, executing a single statement at a time
- *Step Out Of* a function, executing a single statement at a time
- *Quick Watch* a selected variable

Execution is continued by selecting the *Debug, Go* menu option. When the analysis is complete the debug process returns to its initial state.

The debug process may be terminated at any time by selecting the *Debug, End* menu option.

### **Importing and Exporting Data**

LccWare provides extensive import and export facilities allowing users to transfer project data to and from Microsoft Access compatible databases, Microsoft Excel worksheets and text files.

The lccWare project file is based on a database structure in which each database is composed of tables. Each table is composed of records and records are made up of fields. For instance the lccWare project database has tables such as the Cost Nodes table which holds information on the cost nodes in the project. The Constant Global Variables table contains records for each constant global variable and these records contain fields called Name, Description and Default Value. An Excel workbook is equivalent to the database and contains worksheets that correspond to tables. The worksheets are arranged in tabular form with rows corresponding to records that contain fields. A text file corresponds to a table. The file contains records that are composed of fields.

Both the import and export features are accessed from options on the lccWare *File* menu.

The user should choose the *File, Export* menu option to begin the process of specifying the data to export. The Export program will then appear.

The user should choose the *File, Import* menu option to begin the process of specifying the data to import. The Import program will then appear.

The Import and Export programs are generic facilities providing import and export facilities for other compatible applications. The detailed description for each program makes no explicit reference to lccWare, referring instead to the 'application'.

## Security

By default lccWare applies no security restrictions to the project databases it creates. Any user may create and access a lccWare project. In fact if no password has been set for the administrative user *admin* then the normal user will be unaware of the security mechanism.

To activate lccWare security the */secure* flag must be added to the lccWare command line. This will normally be done by appending */secure* to the *Target* property of the existing lccWare shortcut in the lccWare program group, or by creating a new shortcut with the */secure* flag set.

When the security mechanism is activated all project databases have security information appended. This information determines what operations individual users can perform on the lccWare project. When starting lccWare in secure mode the user is presented with a logon dialog requesting the user name and password. A lccWare user may belong to one of four security groups, viz:

- Admins                      This group may create and delete users, change passwords, create lccWare projects and run analyses. There is only one user belonging to this group, the *Admin* user.
- Design lccWare            This group may create lccWare projects and run analyses.
- Modify lccWare Data      This group may modify lccWare project data (but not add cost nodes or functions) and run analyses.
- Read lccWare Data        This group may read lccWare project data and run analyses.

The permissions assigned to each of these groups are described in more detail in Appendix V.

On installation the security system has four users, *admin* (belonging to the *Admins* group), *design* (belonging to the *Design lccWare* group), *modify* (belonging to the *Modify lccWare Data* group), and *read* (belonging to the *Read lccWare Data* group). The passwords are not set for these users.

It should be noted that if the password is set for the *admin* user then even if the */secure* command line flag is not set then the lccWare user will be prompted for a user name and password. In this case lccWare will restrict access to secured projects according to the group membership of the user (in exactly the same manner as if the */secure* flag was set).

The lccWare security system is based on the same system as Microsoft Access. Thus users can not modify or view the database in Microsoft Access (or other products based on the Jet database engine) without the correct permissions.

The lccWare security file (*system.mdw*) is installed in the same directory as the program executable. It is important that this file is protected from being deleted or overwritten by unauthorised users. This is normally achieved by using the operating system file permissions.