SEED-Layout Reference Manual

Ulrich Flemming, Sheng-Fen Chien

School of Architecture and
Institute for Complex Engineered Systems
Carnegie Mellon University
Pittsburgh, PA 15213

9/20/98

1Introduction.....1
1.1 About this Manual 1
1.2 Technical Note 1
2Operations - Overview.....3
3Design Window.....5
3.1 File Menu 6
Load Problem ... 6
Save Problem 6
Save Problem As ... 6
Open Layout ... 7
Save Layout ... 7
Save MDS File 7
Finalize+Save Layout ... 7
Quit 8
3.2 Edit Menu 8
Edit Unit 8
Change Function 8
Grid/Units 9
3.3 View Menu 9
Access Paths 9
Superlayout 9
Zoom Out 10
Zoom In 10
Actual Size 10
3.4 Window Menu 10
Layout Problem 10
Constituent Hierarchy 10
Layout Tree 10
Evaluation Result 11
Layout Problem Manager 11
3.5 Layout Menu 11
Add Unit 11
Remove Unit 12
Generate ... 12
Expand Unit 13
Display ... 13
Goto ... 13
Delete Layout 13
3.6 Problem Menu 14
New 14
Activate Problem 14
Aggregate 14
Disaggregate 15
Goto Superproblem 16
Goto Subproblem 16
Goto Last Problem 16
Goto Next Problem 16
Save Document File 16
3.7 Evaluation Menu 17
3.8 Help Menu 17
3.9 Command Bar 17
Add Unit 17
Remove Unit 18
Edit Unit 18
Change Function 18

Aggregate 19
Disaggregate 20
Expand Unit 20
Delete State 21
Finalize State 21
3.10 Generation Buttons 22
3.11 View Control Buttons 22
3.12 Layout Navigation Buttons 23
3.13 Unplaced Units Dialog Box 23
4File Dialog Boxes 25
4.1 File Menu 25
4.2 Directories Menu 25
4.3 Options Menu 25
Edit files 25
Edit directories 26
Hidden files 26
Create directory 26
4.4 Display Field 26
4.5 Other Features 26
Directory button 26
Name field 26
Filter field 26
Update button 26
5Layout Problem Window.....27
5.1 Create an Initial Layout Problem 28
5.2 File Menu 28
Save Problem 28
Save Problem As ... 28
Close 28
5.3 Edit Menu 29
Add Top FU ... 29
Add Top FU from Library ... 29
Add Constituent ... 29
Add Constituent from FU Library ... 30
Share Constituent ... 30
Add Required Relation ... 31
5.4 View Menu 31
5.5 Window Menu 31
5.6 Command Bar 31
Add Constituent 31
Add Req. Relation 32
Delete Req. Relations 32
Navigation Buttons 33
5.7 Context Field 33
5.8 Required Relations Field 34
Adjacencies 34
Alternative Adjacencies 35
Directions 35
Distances 36
Accessibilities 36
5.9 Functional Unit Field 36
Occupancy 37
Constituents 37
Value Constraint 37
6Functional Unit Window 38

6.1 File Menu 38
Save FU ... 38
Close 38
6.2 Edit Menu 39
Add Constituent ... 39
Add Constituent from FU Library ... 39
Share Constituent ... 40
6.3 View Menu 40
Occupancy 40
Constituents 40
Value Constraint 40
6.4 Occupancy Field 40
6.5 Constituents Field 41
6.6 Value Constraints Field 41
7Layout Problem Manager 42
7.1 File Menu 42
Refresh 42
Close 42
7.2 Problem Menu 42
Activate 42
Remove 42
7.3 Display Field 42
8Constituent Hierarchy Window.....43
8.1 Edit a Functional Unit 43
Open 44
Add Constituent 44
Add Constituent from FU Library 44
8.2 File Menu 45
Refresh 45
Close 45
8.3 Layout Menu 45
LeftRight 45
TopDown 45
Indented 45
9Layout Tree Window 46
9.1 File Menu 46
Refresh 46
Close 46
9.2 Layout Menu 46
LeftRight 46
TopDown 46
Indented 47
10Evaluation Result Window.....48
Appendix A .....
Index 55

1 Introduction

1.1 About this Manual

This reference manual provides a complete description of the SEED-Layout interface and the operations users may trigger through the interface. Readers are assumed to be somewhat familiar with SEED-Layout and the concepts on which it is based. They should specifically understand the following concepts, which are fundamental for the way in which SEED-Layout structures a layout task:

Layout Problem and its components
- **Context**
- **Functional Unit** and Functional Unit **constituents**

Layout and its components
- **Design Unit**
- **Wall**

SEED-Layout **Design Space.**

These terms are capitalized in this manual even if strictly speaking, they are not names. Consult the *SEED - Fundamental Concepts* manual for an introduction to these concepts.

This manual, then, is not a tutorial. It is meant to serve as reference for users who are working with SEED-Layout and want find out from where a certain command can be triggered or how a certain operation works. The individual entries are therefore short and largely self-contained, and we tried to avoid cross-references whenever this did not result in the repetition of large chunks of text and figures.

This manual groups descriptions of the individual features of the interface according to the windows to which the features belong. For example, the commands or items available in the File menu of the Design Window can be found in the Design Window section and File Menu subsection. Thus, features that are near each other in the interface are described in the same section or subsection. This also results in somewhat thematic or functional groupings as the individual windows group features by-and-large according to shared functionalities.

Readers who are not helped by this grouping because they do not remember exactly from where a command can be triggered or where a certain functionality resides are referred to the index. They may also consult the next section, which gives an overview of the functionality of SEED-Layout in terms of the individual tasks it supports and indicates in each case where the present manual describes the respective operation.

1.2 Technical Note

SEED-Layout is based on an object-oriented design. It is written in C++ under the ET++ application framework. Users need not be aware of this at all. But those users who are familiar with object-oriented design may find the following remarks useful because they may allow these users to develop a deeper understanding of how SEED-Layout works, or to develop a more effective ‘mental model’ of its operations.

Objects have states, which are represented by instance variables or attributes. The actions taken or operations executed by an object-oriented program create or delete objects or change the state of existing objects by changing some of their attributes. Many of the view commands available in SEED-Layout (such as Open a Functional Unit) display objects in specific states. If one abstracts from the graphics of the display, the screen image comes close to what is actually stored internally.

Attributes with simple values, like numbers or character strings, can be overwritten in the interface (subject to some limitations); in this case, the user affects directly a state change of the object involved in the operation.

But the objects SEED-Layout creates are normally not independent; they are in the vast majority of cases complex configurations or compositions of linked objects, where the links are represented as relational attributes. Changing these attributes changes the underlying configuration and may trigger updates and notifications of other objects. It is for this reason that users of SEED-Layout have fewer opportunities to change relational attributes directly. When they are allowed to do this, SEED-Layout provides additional support by following up with the needed updates. But since SEED-layout intends to encourage design exploration, it retains configuration states before they were modified. When it changes the state of a relational attribute, it does not modify the existing configuration; rather, it makes a copy of it and adjusts the relations *in the copy* as required by the action. The old configuration remains available and can be revisited, unless the user explicitly deletes it. Knowing this, users will be able to predict when an action causes SEED-Layout to create copies of object configurations or when it changes the state of an existing object.

An exception from this rule is made only when it would result in an obvious overkill; these exceptions are identified in the present manual in the context of the action where they apply.

2 Operations - Overview

This section gives a brief overview of the individual tasks supported by SEED-Layout. It indicates in each case the section in the present manual that describes the interface feature from where a task can be executed. Note that in many cases, the same action can be triggered in multiple ways.

Layout Problem Operations

Display windows:

- Active Layout Problem (Section 3.4.1 on page 10, Section 5 on page 27)
- Constituent hierarchy (Section 3.4.2 on page 10, Section 8 on page 43)
- Layout Problem manager (Section 3.4.5 on page 11, Section 7 on page 42)

Creation/activation:

- Create an initial Layout Problem (Section 5.1 on page 28)
- Create a new Layout Problem (Section 3.6.1 on page 14)
- Open a saved Layout Problem (Section 3.1.1 on page 6)
- Revisit or activate a Layout Problem (Section 3.6.2 on page 14, Section 3.6.2 on page 14, Section 3.6.6 on page 16, Section 3.6.7 on page 16, Section 3.6.8 on page 16)

View parts of a Layout Problem:

- Context (Section 5.4 on page 31, Section 5.7 on page 33)
- Top Functional Unit (Section 5.4 on page 31, Section 5.9 on page 36)
- Required relations(Section 5.4 on page 31, Section 5.8 on page 34)

Editing/modifying a Layout Problem:

- Add Functional Unit constituent (Section 5.3.1 on page 29, Section 5.3.3 on page 29, Section 5.6.1 on page 31, Section 6.2.1 on page 39, Section 6.2.2 on page 39, Section 8.1.2 on page 44, Section 8.1.3 on page 44)
- Share a constituent as a vertical zone (Section 5.3.5 on page 30, Section 6.2.3 on page 40)
- Edit attributes of a Functional Unit (Section 6 on page 38, Section 8.1 on page 43)
- Aggregate/disaggregate constituents: this operation is currently not directly supported; it can be done indirectly by aggregating/disaggregationg Design Units (Section 3.6.3 on page 14, Section 3.6.4 on page 15)

Save in files:

- Save active Layout Problem (Section 3.1.2 on page 6, Section 5.2.1 on page 28)
- Save a copy of the active Layout Problem under a different name - save as (Section 3.1.3 on page 6, Section 5.2.2 on page 28)
- Save Functional Unit in the Functional Unit library (Section 6.1.1 on page 38)
- Set input/output units for saved document file in .html format (Section 3.2.3 on page 9)

Layout Operations

Display options in Design Window:

- Access paths (Section 3.3.1 on page 9)
- Superlayout (Section 3.3.2 on page 9, Section 3.5.5 on page 13)
- Zoom in (Section 3.3.4 on page 10, Section 3.11 on page 22)
- Zoom out (Section 3.3.3 on page 10, Section 3.11 on page 22)
- Actual size/adjust scale (Section 3.3.5 on page 10, Section 3.11 on page 22)

Display windows:

- Display Layout Tree (Section 3.4.3 on page 10, Section 9 on page 46)
- Evaluation results (Section 3.4.4 on page 11, Section 10 on page 48)

Open file:

- Open a saved Layout (Section 3.1.4 on page 7

Layout generation/modification:

- Add a Design Unit under user control (Section 3.5.1 on page 11, Section 3.9.1 on page 17)
- Change order of allocation (Section 3.13 on page 23)
- Generate child/sibling layout (Section 3.5.3 on page 12, Section 3.10 on page 22)
- Remove a Design Unit (Section 3.5.2 on page 12, Section 3.9.2 on page 18)
- Edit the dimensional attributes of a Design Unit (Section 3.2.1 on page 8, Section 3.9.3 on page 18)
- Change the function of a Design Unit (Section 3.2.2 on page 8, Section 3.9.4 on page 18)
- Expand a Design Unit (Section 3.5.4 on page 13, Section 3.9.7 on page 20)
- Aggregate Design Units (Section 3.6.3 on page 14, Section 3.9.5 on page 19)
- Disaggregate a Design Unit (Section 3.6.4 on page 15, Section 3.9.6 on page 20)
- Finalize the active Layout (Section 3.9.9 on page 21)
- Delete the active Layout (Section 3.5.7 on page 13, Section 3.9.8 on page 21)

Navigation/activation:

- Goto parent/sibling/child Layout (Section 3.5.6 on page 13, Section 3.12 on page 23)
- Goto any Layout in layout Tree (Section 9 on page 46)

Save in files:

- Set input/output/grid units for saved layout files (Section 3.2.3 on page 9)
- Save the active Layout (Section 3.1.5 on page 7)
- Save an MDS file (Section 3.1.5 on page 7)
- Save a 3D configuration (Section 3.1.7 on page 7)

3 Design Window

The Design Window is the primary window in SEED-Layout. It is the equivalent of what is called the “Document Window” in other applications. We do not use this name because strictly speaking, users of SEED-Layout do not produce “documents” like drawings or reports. They generate design representations inside the computer that can be used in various ways. The graphics in the Design Window only serve to display this information in a convenient form.

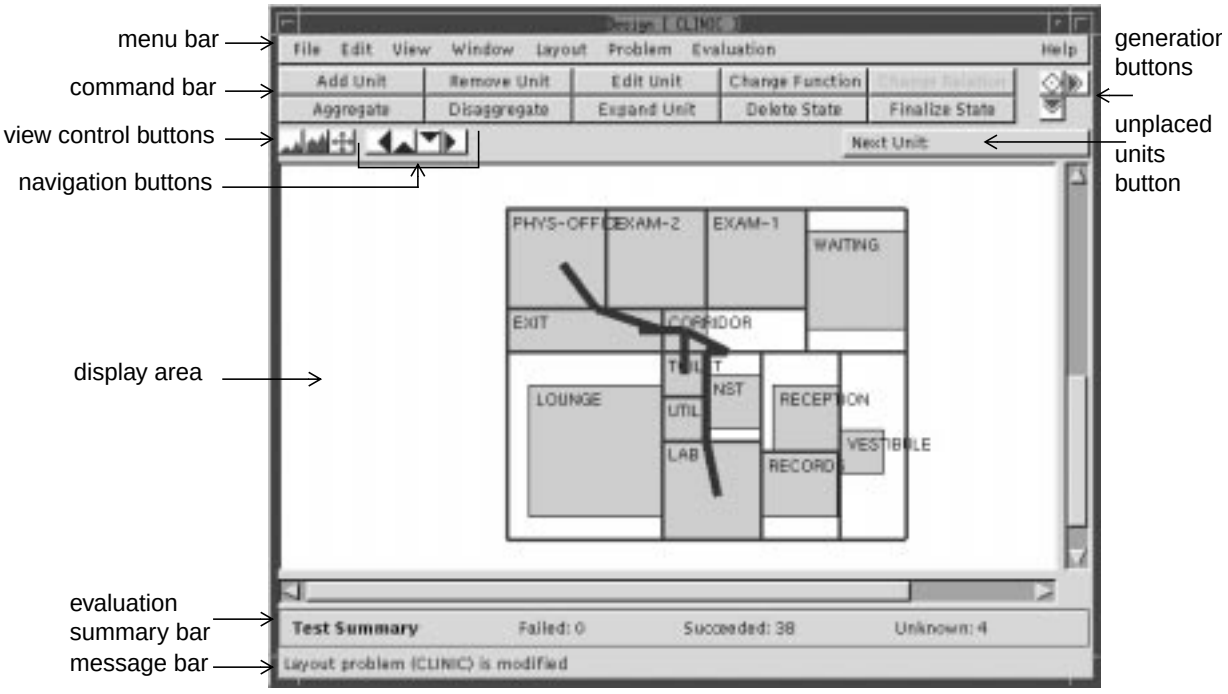


FIGURE 1. Design Window

Figure 1 shows the main components of a Design Window:

- The central display area shows the Layout that is currently active.
- The menu bar contains pull-down menus containing sub-menus, items or options.
- The command bar contains two rows of command buttons that allow the user to trigger the execution of specific design tasks.
- Generation buttons at the right end of the command bar trigger the addition of design units in a more automated mode.
- View control buttons allow users to zoom in or out or change the display scale.
- Navigation buttons allow the user to activate Layouts that have been generated before and can easily be “reached” from the active Layout.
- The unplaced units menu can be used to change the order in which design units are generated.

- The test summary bar displays a summary of the last evaluation of the active Layout: number of requirements failed; number of requirements satisfied; and number of requirements that cannot be evaluated because of insufficient information.
- The message bar displays system messages.

The following sections explain the functionality of these parts in greater detail.

3.1 File Menu

3.1.1 Load Problem ...

This command loads an existing Layout Problem from a file into memory and makes it the active problem.

Steps:

1. **Select the Load Problem... option from the File menu. If a Layout Problem is currently active and has been modified, you are prompted to save or ignore the modifications made to this problem.**
2. **SEED-Layout opens a file dialog box displaying the available Layout Problems in the default directory. You may navigate through the directories to open another one.**
3. **From the file dialog box, select the desired problem and click Open. This closes the file dialog box. If you requested the modifications of the active Layout Problem to be saved in step 1, SEED-Layout saves the active problem and closes it. SEED-Layout loads the selected problem, activates it and displays an initial Layout containing a Design Unit that allocates the first Constituent of the top Functional Unit in the problem.**

3.1.2 Save Problem

This command saves the active Layout (sub)Problem in the file from where it was read.

Caution: The top Functional Unit of the active Layout Problem will be the top Functional Unit overall in the saved problem, and the current Context will be the Context overall independent of where the active problem resides in the current problem hierarchy. That is, if the user is currently working on a subproblem, everything higher-up will be lost. To avoid this problem, back up to the highest level Layout Problem before saving it.

3.1.3 Save Problem As ...

This command saves the active Layout Problem as an independent Layout Problem. That is, the top Functional Unit of the active problem will be the top Functional Unit overall in the saved problem, and the current Context will be the Context overall independent of where the active problem resides in the current problem hierarchy.

Steps:

1. **Select the Save Problem As ... option from the File menu. This opens a file dialog box displaying the available problems in the default directory. You may navigate through the directories to open another one.**
2. **Type name of file in which the problem should be saved in the Save problem as name field or select a file in the display area to be overwritten.**
3. **Click Save to complete the action or Cancel to abort it. This closes the file dialog box.**

3.1.4 Open Layout ...

This command opens a saved Layout and makes it the active Layout. It deletes all other Layouts in the active Design Space.

Steps:

1. **Select the Open Layout ... option from the File menu. This opens a file dialog box displaying the available Layouts in the default directory. You may navigate through the directories to open another one.**
2. **From the file dialog window, select the desired Layout and click Open. This closes the file dialog box.**
3. **SEED-Layout displays the opened Layout as the active Layout in the Design Window.**

Note that a saved Layout saves with each design unit the name of an associated Functional Unit, provided this association existed at the time when the Layout was saved. When SEED-Layout opens a Layout, it associates each design unit in this Layout with a Functional Unit in the active Layout Problem that has the same name as the saved one. If the saved name cannot be found in the active Layout Problem, the respective design unit will have no associated Functional Unit. Users may associate with these design units Functional Units from the active Layout Problem by using the change Functional Unit command.

3.1.5 Save Layout ...

This command saves the active Layout in a file with extension .loos, from where it can be opened again in this or another SEED-Layout session.

Steps:

1. **Select the Save Layout ... option from the File menu. This opens a file dialog box displaying the available Layout files in the default directory. You may navigate through the directories to open another one.**
2. **Type name of file in which the Layout should be saved in the Save Layout As name field or select a file in the display field to be overwritten.**
3. **Click Save to complete the action or Cancel to abort it. This closes the file dialog box.**

3.1.6 Save MDS File

This command saves the active Layout through all levels in the problem hierarchy in an MDS file with extension .slo. You must finalize the active Layout through all levels before executing this command (see Section 3.9.9 on page 21).

Steps:

1. **(Optional) Set the appropriate units and grid (see Section 3.2.3 on page 9).**
2. **Select the Save MDS File... option from the File menu. This opens a file dialog box displaying the available Layout files in the default directory. You may navigate through the directories to open another one.**
3. **Type name of file in which the Layout should be saved in the Save Layout As name field or select a file in the display field to be overwritten.**
4. **Click Save to complete the action or Cancel to abort it. This closes the file dialog box.**

3.1.7 Finalize+Save Layout ...

This command finalizes the active Layout at all levels and saves it in a form that allows a 3-dimensional display using OpenGL.

Steps:

1. **Select the Finalize+Save Layout option from the File menu. This opens a file dialog box displaying the available Layout files in the default directory. You may navigate through the directories to open another one.**
2. **Type name of file in which the Layout should be saved in the Save Layout As name field and select the .oms extension.**
3. **Click Save. This closes the file dialog box.**

3.1.8 Quit

Selecting this option closes all SEED-Layout windows and exits SEED-Layout.

3.2 Edit Menu

This menu contains commands that change the attributes of certain objects. None of the commands results in the creation of a new Layout or alters in any other way a hierarchy in the current design space.

3.2.1 Edit Unit

You may use this command to change directly the dimensional attributes of a Design Unit in the active Layout.

Steps:

1. **Select a Design Unit in the active Layout.**
2. **Select the Edit Unit option in the Layout menu. SEED-Layout opens the Edit Design Unit Window (see Figure 2). Overwrite any coordinate bound you wish to change, but note that lower bounds can only increase and upper bounds can only decrease.**
3. **Click COMMIT or CANCEL.**
4. **If you clicked COMMIT, SEED-Layout tries to resize the active Layout. If it succeeds, it refreshes the display; otherwise, it aborts the command and displays an error message.**

lower/upper bound of lower x-coordinate
lower/upper bound of upper x-coordinate
lower/upper bound of lower y-coordinate
lower/upper bound of upper y-coordinate

FIGURE 2. Edit Design Unit Window

3.2.2 Change Function

If two Design Units have been selected, SEED-Layout tries to swap the associated Functional Units in the active Layout. If only one Design Unit has been selected, SEED-Layout tries to replace the Functional Unit associated with this Design Unit in the active

Layout with the next unplaced Functional Unit. If it succeeds, SEED-Layout resizes the active Layout.

Steps:

1. **Select one Design Unit in the active Layout.**
2. **Select the Change Function option. If replacing the Functional Unit associated with the selected Design Unit by the next Unplaced Unit is feasible, SEED-Layout refreshes the display of the active Layout and puts the replaced Unit back in the Unplaced Units list (at the end); otherwise, it aborts the operation and displays an error message.**

or

1. **Select two Design Units in the active Layout.**
2. **Select the Change Function option in the Layout menu. If swapping the Functional Units is feasible, SEED-Layout refreshes the display of the active Layout; otherwise, it aborts the operation and displays an error message.**

Note that this commend does not result in a new active Layout.

3.2.3 Grid/Units

FIGURE 3. Grid/Units Window

Use this option to set

- the project unit used in the current Layout Problem in the Project Unit pop-up menu.
- the unit to be used in the next MDS file in the Output Unit pop-up menu.
- the snap grid to be used in the next MDS file in the Grid Size field. The size of the basic grid unit must be expressed in integer multiples of the output unit.
  Example: If the output unit is *mm* and the snap grid should have 30 *cm* intervals, set output unit to *mm* and grid size to 300.

3.3 View Menu

3.3.1 Access Paths

This is a toggle button that turns the display of access paths in the Design Window on or off (see Figure 1 for an example of access path displays).

3.3.2 Superlayout

This is a toggle button that turns the display of superlayouts in the Design Window on or off. The superlayout is the Layout of which the active Layout is a sublayout.

In the current implementation, superlayouts are displayed only for the next higher level with respect to the active Layout; Figure 4 shows an example. Note that before finalization, sublayouts usually do not fit exactly into their superlayouts.

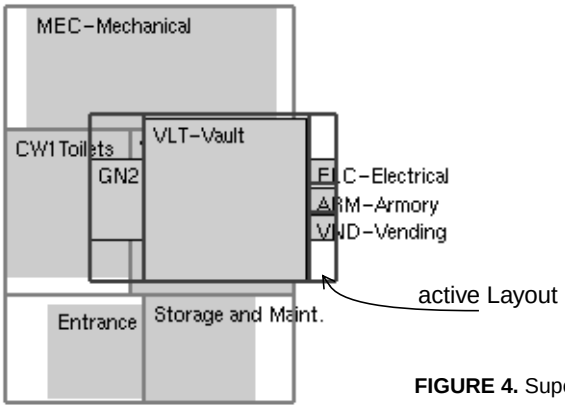


FIGURE 4. SuperLayout display - example

3.3.3 Zoom Out

This option reduces the magnification of the display in the Design Window by a factor of approximately .83 (1/1.20).

3.3.4 Zoom In

This option increases the magnification of the display in the Design Window by 20% or by a factor of 1.20.

3.3.5 Actual Size

This option rescales the display in the Design Window to the original size.

3.4 Window Menu

This menu contains options to open supplementary windows.

3.4.1 Layout Problem

Use this option to open the Layout Problem Window, which allows you to inspect and edit the settings in the active Layout Problem; see Section 5 on page 27 for details.

3.4.2 Constituent Hierarchy

Use this option to open the Constituent Hierarchy Window, which displays the Constituents of the top Functional Unit of the active Layout Problem down through the entire structure; see Section 8 on page 43 for details.

3.4.3 Layout Tree

Use this option to open the Layout Tree Window, which displays the parent-child relations between all Layouts generated so far for the active Layout Problem; see Section 9 on page 46 for details.

3.4.4 Evaluation Result

Use this option to open the Evaluation Result Window, which displays detailed evaluations of the Layouts displayed in the Design Window; see Section 10 for a detailed description.

3.4.5 Layout Problem Manager

Use this option to open the Layout Problem Manager, a window that displays the problems, subproblems and problem versions generated so far. The window can be used to reactivate any one of the problems; see Section 7 on page 42 for details.

3.5 Layout Menu

The Layout menu contains the commands that ask SEED-Layout to create Layouts from existing ones. It also contains commands to reactivate the Layouts generated so far for the active Layout Problem.

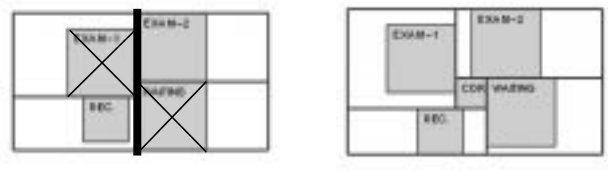
3.5.1 Add Unit

Use this command when you know exactly where the next Functional Unit should be allocated in relation to the existing Walls and Design Units in the active Layout. We call this operation also “allocation or placement under designer control.”

Steps:

1. **Select a Wall in the active Layout and one or two Design Units adjacent to that Wall.**
2. **(Optional) Change the next Functional Unit to be placed.**
3. **Select the Add Unit option from the Layout menu. If not enough elements are selected, if the selected Design Units do not border the selected Wall, or if the planned insertion is topologically impossible, SEED-Layout displays an error message and aborts the operation.**
4. **SEED-Layout allocates the selected Functional Unit in an area created by moving the selected Design Unit(s) away from the selected Wall. If this placement is not feasible, an error message is displayed (in rare cases, the entire operation is aborted). The new Layout becomes the active Layout and is displayed in the Design Window.**

Note that SEED-Layout allocates the Functional Unit in a copy of the active Layout and makes this copy the new active Layout. That is, the former active Layout is not destroyed and can be revisited. As a general principle, the system implements a requested change in the active Layout in a *copy* of that Layout whenever the change implies a structural change.



a b

FIGURE 5. a) a Layout with a selected Wall and two selected Design Units at opposite sides; b) the result of adding a new Design Unit with this selection

The results of this operation vary significantly if two Design Units are adjacent to the selected Wall from the same or opposing sides. In the first case, the operation is straightforward: the Design Units appear to get “pushed away” from the Wall so that the new Design Unit can be inserted in the area created in this way. If the Design Units border the Wall from opposite sides, SEED-Layout splits the Wall and creates a “pinwheel” or “spiral” configuration (see Figure 5 for an example). This can be quite unexpected; in some cases, SEED-Layout generates two new Layout s, one with a clockwise turning and one with a counter-clockwise turning pinwheel.

3.5.2 Remove Unit

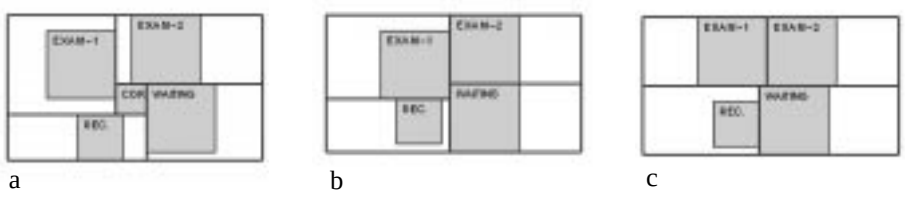
This command causes SEED-Layout to remove a Design Unit from the active Layout and to make the associated Functional Unit available for reallocation.

Steps:

1. **Select a Design Unit in the active Layout.**
2. **Select the Remove Unit option from the Layout menu. If the Design Unit is a pinwheel, SEED-Layout prompts you for further information to remove ambiguities:**

- **horizontal: collapse the pinwheel into a horizontal Wall**
- **vertical: collapse the pinwheel into a vertical Wall**
- **generate: determine which possibility is feasible and generate the first feasible possibility.**

3. **If the removal is feasible, SEED-Layout removes the selected Design Unit and puts the associated Functional Unit back on the Unplaced Units list. The new Layout becomes the active Layout and is displayed in the Design Window.**



a b c

FIGURE 6. Removing the corridor in (a) with option ‘vertical’ (b) or ‘horizontal’ (c)

Note that SEED-Layout removes the selected Design Unit in a copy of the active Layout and makes the copy the new active Layout. That is, the former active Layout is not destroyed and can be revisited.

3.5.3 Generate ...

This item opens a submenu offering two options:

- **Child state**: Selecting this option makes SEED-Layout attempt to generate the next feasible child Layout of the active Layout. If such a Layout is found, it becomes the active Layout and is displayed in the Design Window. Otherwise, SEED-Layout aborts the operation and displays an error message.
- **Sibling state**: Selecting this option makes SEED-Layout attempt to generate the next feasible sibling Layout of the active Layout, that is, the next child of the active Layout’s parent Layout. If such a Layout is found, it becomes the active Layout and is displayed in the Design Window. Otherwise, SEED-Layout aborts the operation and displays an error message.

3.5.4 Expand Unit

Use this option to expand a Design Unit, that is, to create a subproblem to allocate the Constituents of the Functional Unit associated with the Design Unit inside its boundary.

Steps:

1. **Select a Design Unit.**
2. **Select the Expand Unit option in the Generation menu. If the selected Design Unit has been expanded before, SEED-Layout displays an error message and aborts the operation.**
3. **SEED-Layout creates a subproblem and activates it. Its top Functional Unit is the Functional Unit associated with the Design Unit selected in step 1. Its Context is the boundary of that Design Unit. The initial Layout in the corresponding subspace is a Layout allocating the first Constituent of the top Functional Unit. It is displayed in the Design Window.**

Note that if you return to the superproblem and generate child Layouts of the Layout whose Design Unit was expanded, the subspace will be a subspace of this Design Unit in all child Layouts containing this Design Unit (the Design Unit may disappear from a child Layout because the user may have deleted it with the Remove Unit command).

3.5.5 Display ...

This option opens a submenu offering two options:

- **Sublayout**: This option is currently not implemented.
- **Superlayout**: Selecting this option makes SEED-Layout display the superlayout of the active Layout, if it exists. If the active Layout Problem is at the root of the Layout Problem hierarchy, no superlayout can exist, and SEED-Layout displays an error message.

3.5.6 Goto ...

This option opens a submenu offering the following options to reactivate Layouts “near” the active Layout:

- **Parent State**: This option reactivates the Layout from which the active Layout was generated, if it exists.
- **Previous Sibling State**: This option reactivates the last sibling Layout generated before the active Layout was generated, if it exists.
- **Next Sibling State**: This option reactivates the first sibling Layout generated after the active Layout was generated, if it exists.
- **Child State**: This option reactivates the first child Layout of the active Layout.

Attempts to reactivate a non-existing Layout result in error messages. Note that these commands do not generate Layouts.

3.5.7 Delete Layout

Selecting this option triggers the deletion of the active Layout and all Layouts generated from it. Use this option if you want to make more memory available by deleting Layouts that are no longer of interest to you.

Steps:

1. **Select the Delete State option in the Layout menu.**

2. **SEED-Layout deletes the active state and all Layouts generated from it. It makes the parent of the active Layout the active Layout and refreshes the Design Window.**

3.6 Problem Menu

This menu allows you to create a new or initial overall Layout Problem 'on-the-fly' and to revisit Layout Problems that were created before in the current Design Space.

3.6.1 New

This command closes the currently loaded Layout problem and creates a new one. It is disabled if no active Layout problem exists.

Steps:

Select the New option from the File menu. If the currently loaded Layout Problem has been modified, you are prompted to save or ignore the modifications made to the problem. The system closes the current problem, then opens a new Design Window.

Caution: The new Layout Problem is empty and does not have necessary information to constitute a valid problem when it is first created. If you save the empty problem, it cannot be reloaded. You should always open the Layout Problem window and use commands in that window to input/edit the content of this new Layout Problem before saving it (see Section 5 on page 27).

3.6.2 Activate Problem

Use this option to activate an initial Layout Problem that you created from scratch (see Section 5.1 on page 28). This option is disabled if an active Layout problem exists.

If you wish to replace the currently loaded Layout Problem with a new one created from scratch, use the New option (see Section 3.6.1 on page 14).

3.6.3 Aggregate

This command can be used to aggregate selected Design Units in the active Layout into a single Design Unit. As a side effect of this operation, SEED-Layout creates a variant of the active Layout Problem with a Constituent hierarchy consistent with the aggregation.

Steps:

1. **Select four Walls forming a rectangle in the active Layout. Make sure that the selected Walls enclose more than one Design Unit, but not the entire Layout.**
2. **Select the Aggregate option from the Layout menu. If the Walls are not properly formed, SEED-Layout displays an error message and aborts the operation.**
3. **SEED-Layout prompts you for the name and class of a new Functional Unit.**
4. **SEED-Layout creates a variant of the active Layout Problem and a variant Design Space. The initial Layout in this space is the aggregated Layout, in which the Design Units inside the selected rectangle are replaced by a single Design Unit associated with the Functional Unit created in step 2. This Layout is the new active Layout and displayed in the Design Window.**

In executing this command, SEED-Layout performs a complex sequence of operations in the background:

1. Call the top Functional Unit of the active Layout *A*. SEED-Layout creates a variant of the active Layout Problem and a copy of *A*. Call this copy *B*. SEED-Layout makes *B* the top Functional Unit of the Layout Problem variant. The Constituents of *B* are the Constituents of *A* except for the Functional Units associated with the Design Units inside the selected rectangle; these units are replaced by the Functional Unit created in step 2; call this Unit *N*. The replaced Functional Units become the Constituents of *N* (see Figure 7).

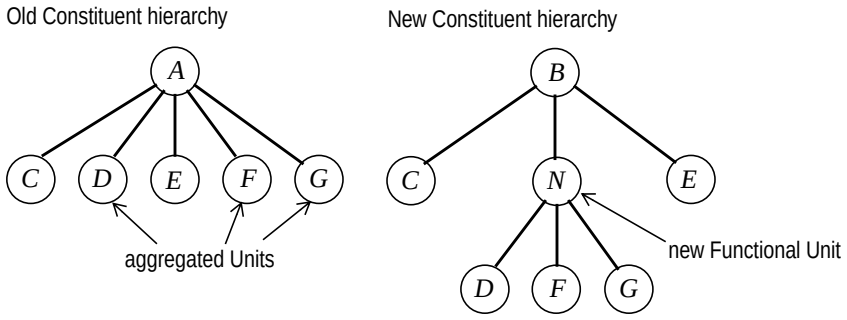


FIGURE 7. Functional Unit aggregation - example

2. SEED-Layout creates a copy of the active Layout and replaces the Design Units inside the selected rectangle with a single, new Design Unit; call this Design Unit *d*. It becomes associated with *N*. The new Layout becomes the first and active Layout in a new Design Space
3. SEED-Layout creates a subspace of the new Design Space and an associated Layout Problem whose top Functional Unit is *N* and whose Context is given by the geometry of *d*. The first and active Layout in this subspace is the Layout of Design Units inside the initially selected rectangle.

Note that the original Layout Problem and Design Space remain available and can be revisited.

3.6.4 Disaggregate

This command can be used to disaggregate a Design Unit in the active Layout into the Design Units of a sublayout inside this Design Unit. As a side effect of this operation, SEED-Layout creates a variant of the active Layout Problem with a Constituent hierarchy consistent with the disaggregation.

Steps:

1. **Select a Design Unit in the active Layout that has been expanded before.**
2. **Select the Disaggregate option from the Layout menu. If the selected Design Unit has not been expanded before, SEED-Layout displays an error message and aborts the operation.**
3. **SEED-Layout creates a variant of the active Layout Problem and a variant Design Space whose initial Layout is the disaggregated Layout, in which the Design Units inside the selected Design Unit replace this Design Unit. This new Layout becomes the active Layout and is displayed in the Design Window.**

In executing this command, SEED-Layout performs a complex sequence of operations in the background:

1. Call the top Functional Unit in the active Layout Problem *A* and the Functional Unit associated with the selected Design Unit *F*. SEED-Layout creates a variant of the active Layout Problem and a copy of *A*; call this copy *B*. *B* becomes the top unit in the variant Layout Problem. It has the same Constituents as *A* except for *F*, which is replaced by its Constituents. The variant Layout Problem becomes the active one.
2. SEED-Layout creates a copy of the active Layout and replaces the selected Design Unit with the sublayout of Design Units that are the active Layout in the associated subspace; this Layout becomes the first and active Layout in a new Design Space.

Note that the original Layout Problem and Design Space remain available and can be revisited.

3.6.5 Goto Superproblem

Use this option to reactivate the Layout Problem one level up in the hierarchy. SEED-Layout displays an error message if this Layout Problem does not exist. Otherwise, it makes the last active Layout in the associated Design Space the active Layout and displays it in the Design Window.

3.6.6 Goto Subproblem

Use this option to reactivate the first Layout Problem one level down in the hierarchy. SEED-Layout displays an error message if this Layout Problem does not exist. Otherwise, it makes the last active Layout in the associated Design Space the active Layout and displays it in the Design Window.

3.6.7 Goto Last Problem

Use this option to reactivate the last Layout Problem created before the active Layout Problem at the same level in the hierarchy. SEED-Layout displays an error message if this Layout Problem does not exist. Otherwise, it makes the last active Layout in the associated Design Space the active Layout and displays it in the Design Window.

3.6.8 Goto Next Problem

Use this option to reactivate the first Layout Problem created after the active Layout Problem at the same level in the hierarchy. SEED-Layout displays an error message if this Layout Problem does not exist. Otherwise, it makes the last active Layout in the associated Design Space the active Layout and displays it in the Design Window.

3.6.9 Save Document File

Use this option to write an .html file describing the active Layout Problem.

Steps:

1. **Select the Save Document File option in the Problem menu of the Design Window. This opens a file dialog box.**
2. **Enter or select the name of the file. Do not add the .html extension.**
3. **Click Save.**

3.7 Evaluation Menu

This menu will eventually allow users to trigger compute-intensive evaluations or evaluations requiring external software that are not done automatically after each generation step. No such evaluations are currently available.

The **Semper** option opens a window showing the "canned" results of a possible energy evaluation performed with the Semper system currently under development at CMU[1]. This is only meant to give users a feel for how these evaluations may be integrated into SEED-Layout.

3.8 Help Menu

The Help menu and the two options it offers serve right now only as reminders that a full-fledged production version of SEED-Layout would contain an on-line help capability.

3.9 Command Bar

This bar is intended for users who desire to customize the SEED-Layout interface by moving frequently used commands out of the menus where they are hidden so that they can be executed immediately. However, this customization is not available in the current version of SEED-Layout. The command bar of this version simply provides buttons to trigger directly commands that are contained as options in the Layout and Generation menus.

3.9.1 Add Unit

This button allows you to determine exactly where the next Functional Unit is to be allocated in relation to the Design Units and Walls in the active Layout. We call this operation also "allocation or placement under designer control."

Steps:

1. **Select a Wall in the active Layout and one or two Design Units adjacent to that Wall.**
2. **(Optional) Change the next Functional Unit to be placed.**
3. **Click the Add Unit button in the Command Bar. If not enough elements are selected, if the selected Design Units do not border the selected Wall, or if the planned insertion is topologically impossible, SEED-Layout displays an error message and aborts the operation.**
4. **SEED-Layout allocates the selected Functional Unit in an area created by moving the selected Design Unit(s) away from the selected Wall. If this placement is not feasible, an error message is displayed (in rare cases, the entire operation is aborted). The new Layout becomes the active Layout and is displayed in the Design Window.**

Note that SEED-Layout allocates the Functional Unit in a copy of the active Layout and makes this copy the new active Layout. That is, the former active Layout is not destroyed and can be revisited. As a general principle, the system implements a requested change in the active Layout in a *copy* of that Layout whenever the change implies a structural change.

---

1. Mahdavi, A.

The results of this operation vary significantly if two Design Units are adjacent to the selected Wall from the same or opposing sides. In the first case, the operation is straightforward: the Design Units appear to get "pushed away" from the Wall so that the new Design Unit can be inserted in the area created in this way. If the Design Units border the Wall from opposite sides, SEED-Layout splits the Wall and creates a "pinwheel" or "spiral" configuration (see Figure 5 for an example). This can be quite unexpected; in some cases, SEED-Layout generates two new Layouts, one with a clockwise-turning and one with a counterclockwise-turning pinwheel.

3.9.2 Remove Unit

Clicking this button causes SEED-Layout to remove a Design Unit from the active Layout and to make the associated Functional Unit available for reallocation.

Steps:

1. **Select a Design Unit in the active Layout.**
2. **Click the Remove Unit button in the Command Bar. If the Design Unit is a pinwheel, SEED-Layout prompts you for further information to remove ambiguities:**
   - **horizontal: collapse the pinwheel into a horizontal Wall**
   - **vertical: collapse the pinwheel into a vertical Wall**
   - **generate: determine which possibility is feasible and generate the feasible possibilities.**
3. **If the removal is feasible, SEED-Layout removes the selected Design Unit and puts the associated Functional Unit back on the Unplaced Units list. The new Layout becomes the active Layout and is displayed in the Design Window.**

Note that SEED-Layout removes the selected Design Unit in a copy of the active Layout and makes the copy the new active Layout. That is, the former active Layout is not destroyed and can be revisited.

3.9.3 Edit Unit

You may use this button to change directly the dimensional attributes of a Design Unit in the active Layout.

Steps:

1. **Select a Design Unit in the active Layout.**
2. **Click the Edit Unit button in the Command Bar. SEED-Layout opens the Edit Design Unit Window (Figure 2). Overwrite any coordinate bound you wish to change, but note that lower bounds can only increase and upper bounds can only decrease.**
3. **Click COMMIT or CANCEL.**
4. **If you clicked COMMIT, SEED-Layout tries to resize the active Layout. If it succeeds, it refreshes the display; otherwise, it aborts the command and displays an error message.**

3.9.4 Change Function

If two Design Units have been selected, SEED-Layout tries to swap the associated Functional Units in the active Layout. If only one Design Unit has been selected, SEED-Layout tries to replace the Functional Unit associated with this Design Unit in the active Layout with the next unplaced Functional Unit. If it succeeds, SEED-Layout resizes the active Layout.

Steps:

1. **Select one Design Unit in the active Layout.**

2. **Click the Change Function button in the Command Bar. If replacing the associated Functional Unit with the next unplaced Unit is feasible, SEED-Layout refreshes the display of the active Layout and puts the replaced Unit back in the Unplaced Units list (at the end); otherwise, it aborts the operation and displays an error message.**

or

1. **Select two Design Units in the active Layout.**
2. **Click the Change Function button in the Command Bar. If swapping the associated Functional Units is feasible, SEED-Layout refreshes the display of the active Layout; otherwise, it aborts the operation and displays an error message.**

Note that this commend does not result in a new active Layout.

3.9.5 Aggregate

This button can be used to aggregate selected Design Units in the active Layout into a single unit. As a side effect of this operation, SEED-Layout creates a variant of the active Layout Problem with a Constituent hierarchy consistent with the aggregation.

Steps:

1. **Select four Walls forming a rectangle in the active Layout. Make sure that the selected Walls enclose more than one Design Unit, but not the entire Layout.**
2. **Click the Aggregate button in the Command Bar. If the selected Walls are not properly formed, SEED-Layout displays an error message and aborts the operation.**
3. **SEED-Layout prompts you for the name and class of a new Functional Unit.**
4. **SEED-Layout creates a variant of the active Layout Problem and a variant Design Space. The initial Layout in this space is the aggregated Layout, in which the Design Units inside the selected rectangle are replaced by a single Design Unit associated with the Functional Unit created in step 2. This Layout is the new active Layout and displayed in the Design Window.**

In executing this command, SEED-Layout performs a complex sequence of operations in the background:

1. Call the top Functional Unit of the active Layout *A*. SEED-Layout creates a variant of the active Layout Problem and a copy of *A*. Call this copy *B*. SEED-Layout makes *B* the top Functional Unit of the Layout Problem variant. The Constituents of *B* are the Constituents of *A* except for the Functional Units associated with the Design Units inside the selected rectangle; these units are replaced by the Functional Unit created in step 2; call this Unit *N*. The replaced Functional Units become the Constituents of *N* (see Figure 7).
2. SEED-Layout creates a copy of the active Layout and replaces the Design Units inside the selected rectangle with a single, new Design Unit; call this Design Unit *d*. It becomes associated with *N*. The new Layout becomes the first and active state in a new Design Space
3. SEED-Layout creates a subspace of the new Design Space and an associated Layout Problem whose top Functional Unit is *N* and whose Context is given by the geometry of *d*. The first and active Layout in this subspace is the Layout of Design Units inside the initially selected rectangle.

Note that the original Layout Problem and Design Space remain available and can be revisited.

3.9.6 Disaggregate

This button can be used to disaggregate a Design Unit in the active layout into the Design Units of a sublayout inside this Design Unit. As a side effect of this operation, SEED-Layout creates a variant of the active Layout Problem with a Constituent hierarchy consistent with the disaggregation.

Steps:

1. **Select a Design Unit in the active Layout that has been expanded before.**
2. **Click the Disaggregate button in the Command Bar. If the selected Design Unit has not been expanded before, SEED-Layout displays an error message and aborts the operation.**
3. **SEED-Layout creates a variant of the active Layout Problem and a variant Design Space whose initial Layout is the disaggregated Layout, in which the Design Units inside the selected Design Unit replace this Design Unit. This new Layout becomes the active Layout and is displayed in the Design Window.**

In executing this command, SEED-Layout performs a complex sequence of operations in the background:

1. Call the top Functional Unit in the active Layout Problem *A* and the Functional Unit associated with the selected Design Unit *F.* SEED-Layout creates a variant of the active Layout Problem and a copy of *A*; call this copy *B. B* becomes the top unit in the variant Layout Problem. It has the same Constituents as *A* except for *F,* which is replaced by its Constituents. The variant Layout Problem becomes the active one.
2. SEED-Layout creates a copy of the active Layout and replaces the selected Design Unit with the subLayout of Design Units that are the active Layout in the associated subspace; this Layout becomes the first and active Layout in a new Design Space.

Note that the original Layout Problem and Design Space remain available and can be revisited.

3.9.7 Expand Unit

Use this button to “expand” a Design Unit, that is, to create a subproblem to allocate the Constituents of the Functional Unit associated with the Design Unit inside its boundary.

Steps:

1. **Select a Design Unit.**
2. **Click the Expand Unit button in the Command Bar. If the selected Design Unit has been expanded before, SEED-Layout displays an error message and aborts the operation.**
3. **SEED-Layout creates a subproblem and activates it. Its top Functional Unit is the Functional Unit associated with the Design Unit selected in step 1. Its Context is the boundary of that Design Unit. The initial Layout in the corresponding subspace is a Layout allocating the first Constituent of the top Functional Unit. It is displayed in the Design Window.**

Note that if you return to the superproblem and generate child Layouts of the Layout whose Design Unit was expanded, the subspace will be a subspace of this Design Unit in all child Layouts containing this Design Unit (the Design Unit may disappear from a child Layout because you may have deleted it with the Remove Unit command).

3.9.8 Delete State

Clicking this button triggers the deletion of the active Layout and all Layouts generated from it. Use this option if you want to make more memory available by deleting Layouts that no longer interest you.

Steps:

1. **Click the Delete State button in the Command Bar.**
2. **SEED-Layout deletes the active state and all Layouts generated from it. It makes the parent of the active Layout the active Layout and refreshes the Design Window.**

3.9.9 Finalize State

You may use this button to "finalize" the active Layout, that is, to create a Layout with fixed dimensions. This may be needed to supply input to other applications or SEED modules. Specifically, you must trigger this command before generating an MDS-file.

Steps:

1. **Click the Finalize State button in the Command Bar. SEED-Layout opens the Finalize dialog box.**
2. **Select one of the options offered:**

- **Yes: Layouts will be finalized throughout the problem hierarchy. SEED-Layout picks the last active Layout in each subspace and coordinates them so that they fit together, if this is possible at all.**
- **No: SEED-Layout finalizes only the active Layout.**
- **Cancel: Abort operation.**

3. **SEED-Layout creates the finalized Layout (s) and updates the Design Window.**

If you selected the Yes option in step 2, SEED-Layout performs a complex sequence of operations in the background.

1. It visits all subspaces that have created for the currently active Layout Problem and finds the last active Layout in each subspace. This may involver finding the last active version of the subspace if you used the aggregate or disaggregate commands at some time.
2. It creates constraints that assure that the selected sublayouts fit properly into the boundaries of the Design Units they expand. If these constraints cannot be solved, SEED-Layout creates error messages to be displayed at the end of the operation.
3. It instantiates all relational constraints whose Functional Units have been allocated in some active sub(layout) together with all constraints specified by the Functional Units allocated in each sublayout.
4. It tries to solve the resulting systems of constraints. If more than one solution exists, it tries to minimize at the same time the overall area; or, if the layouts allocate massing elements, it tries to minimize the areas occupied by the massing elements.
5. It displays in the Design Window the finalized (sub)layout at the level from where the finalize command was issued. You may visit the other finalized Layouts by navigating through the problem hierarchy (see Section 3.6 on page 14).

Note that SEED-Layout does not finalize the active Layouts themselves, but copies thereof; that is, the non-finalized versions can be reactivated.

But note also that the finalized Layouts are now the active Layouts in the respective Design Spaces. Keep this in mind if you do not quit SEED-Layout soon, but explore further layout options and try to finalize another layout alternative through all levels of the

problem hierarchy. This will not work in all likelihood if a finalized sublayout remains active in some subspace: it now has fixed dimensions and will therefore be hard to fit into a differently dimensioned superlayout. In order to avoid this problem, you will have to revisit each subspace and make sure that the active layout in this space is the one you want to include in the finalization. Some automated support for this process is conceivable, but currently not implemented.

3.10 Generation Buttons

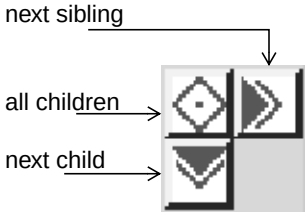


FIGURE 8. Generation buttons

The generation buttons are located at the right end of the command bar (see Figure 8).

Next sibling: Clicking this button makes SEED-Layout attempt to generate the next feasible sibling Layout of the active Layout, that is, the next child of the active Layout's parent Layout. If such a Layout is found, it becomes the active Layout and is displayed in the Design Window. Otherwise, SEED-Layout displays an error message and aborts the operation.

All children: Clicking this button makes SEED-Layout attempt to generate all feasible children of the active Layout. If no feasible child Layout exists, SEED-Layout displays an error message and aborts the operation. Otherwise, the last feasible Layout found becomes the active Layout and is displayed in the Design Window. Use the Layout navigation tools to visit the other feasible Layouts. You may interrupt this operation in the interrupt dialog box that is opened as a result of this commend.

Next child: Clicking this button makes SEED-Layout attempt to generate the next feasible child Layout of the active Layout. If such a Layout is found, it becomes the active Layout and is displayed in the Design Window. Otherwise, SEED-Layout displays an error message and aborts the operation.

Note that if more than one Functional Unit has been selected in the Unplaced Units window, any one of these commands allocates all selected Functional Units in one step.

3.11 View Control Buttons

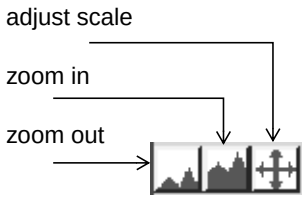


FIGURE 9. View control buttons

The view control buttons are located below the left end of the command bar (see Figure 9).

Adjust scale: Clicking this button makes SEED-Layout attempt to find a more appropriate magnification factor for the display in the Design Window.

Zoom in: Clicking this button increases the magnification of the display in the Design Window by 20% or by a factor of 1.20.

Zoom out: Clicking this button reduces the magnification of the display in the Design Window by a factor of approximately .83 (1/1.20).

3.12 Layout Navigation Buttons

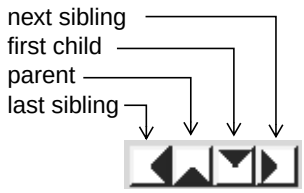


FIGURE 10. Layout navigation buttons

The Layout navigation buttons are located next to the view control buttons (see Figure 10).

Next sibling: Clicking this button reactivates the first sibling Layout generated after the active Layout was generated, provided this sibling exists.

First child: Clicking this button reactivates the first child Layout generated from the active Layout, provided at least one child Layout has been generated before.

Parent: Clicking this button reactivates the Layout from which the active Layout was generated, if this parent Layout exists (that is, the active Layout is not the root Layout of the active Design Space).

Last sibling: Clicking this button reactivates the last sibling Layout generated before the active Layout was generated, provided this sibling exists.

SEED-Layout displays an error message if the Layout to which you wish to return does not exist.

3.13 Unplaced Units Dialog Box

Open the Unplaced Units dialog box from the Next Unit pop-up menu below the Generation Buttons in the Design Window. This box allows you to change the order in which Functional Units are allocated and to select several Functional Units for simultaneous allocation (see Figure 11).

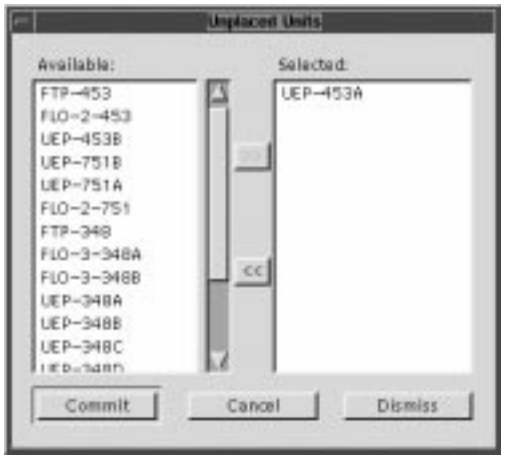


FIGURE 11. Unplaced Units dialog box

The Unplaced Units dialog box is non-modal; can be dismissed by clicking the Dismiss button. The left-hand Available field displays the names of the Functional Unit Constituents in the active Layout Problem that have not been allocated in the active Layout. Move a Constituent to the top of the allocation order as follows:

Steps:

1. **Select the name in the Available field and click the right-pointing arrow button. You will see how the name pops up in the Selected field on the right.**

2. **Repeat this step for multiple selections**

   or

   Remove a selection by selecting the Functional Unit name in the Selected field and clicking the left-pointing arrow button.
3. **Click the Commit button when you are done or Cancel to undo the selections.**
4. **Click Dismiss to close the box.**

4 File Dialog Boxes

Most widgets in the file dialog box are not SEED-Layout specific; they are features provided by the ET++ application framework, which is used not only by SEED-Layout, but also by SEED-Pro and the SEED development environment. The contents displayed in these widgets are therefore not necessarily relevant to SEED-Layout.

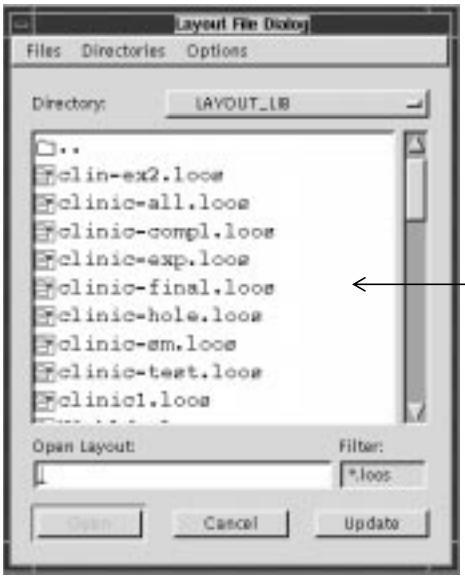


FIGURE 12. Typical file dialog box

4.1 File Menu

The file menu contains the full path names of the files that were selected in any ET++-based application; the files are listed in reverse order (last selected-first in list) and independently of their content.

4.2 Directories Menu

The directories menu contains the full path names of the directories that were selected in any ET++-based application; the directories are listed in reverse order and independently of their content. Double-clicking a name opens this directory and displays the files contained in it in the file selection field.

4.3 Options Menu

4.3.1 Edit files

<to be added>

4.3.2 Edit directories

<to be added>

4.3.3 Hidden files

This is a toggle that controls the display of ‘hidden’ files.

4.3.4 Create directory

4.4 Display Field

Single-clicking a name in the display field selects the respective file and makes the file name appear in the name field below.

Double-clicking a name executes the command that opened the file dialog box. If this was an Open command, the selected file is opened, but only if the file has the appropriate content; for example, opening a file in a dialog box brought up as a result of the Open Layout command will succeed only if the file actually contains a saved Layout. If the command was a Save command, the selected file will be overwritten, but you are prompted explicitly to confirm this.

4.5 Other Features

The Save/Open and Cancel buttons at the bottom of the box work as expected.

4.5.1 Directory button

This is a pop-up button by means of which the path of the current directory can be traversed upwards.

4.5.2 Name field

This field can be used in Open file commands to type-in the name of the desired file. It can be used in Save commands to type-in the name under which a file is to be saved.

4.5.3 Filter field

You may type-in a pattern to be used in selecting the files that are displayed in the display field. For example, the pattern *CLINIC* will restrict the display to files containing the string ‘CLINIC’ in their name. In order to activate a pattern, the Update button must be clicked.

4.5.4 Update button

Clicking this button updates the file selection field using the current filter.

5 *Layout Problem Window*

Figure 13 shows a snapshot of the Layout Problem Window. You open up the Layout Problem Window by selecting the Layout Problem option under the Window menu in the Design Window.

The Layout Problem Window displays the attributes of the active Layout Problem in categories such as Context, Relational Constraints, the associated Functional Unit, and its Constituents. You may expand the display of each category by using, for example, the arrow button in the upper right corner; Figure 13 shows this for the Context category. Navigation buttons in the upper right-hand corner allow you to activate Layout Problems that are 'close' to the one displayed in the associated problem hierarchy. A command bar shows buttons that can be selected to execute tasks needing no further specifications from the designer.

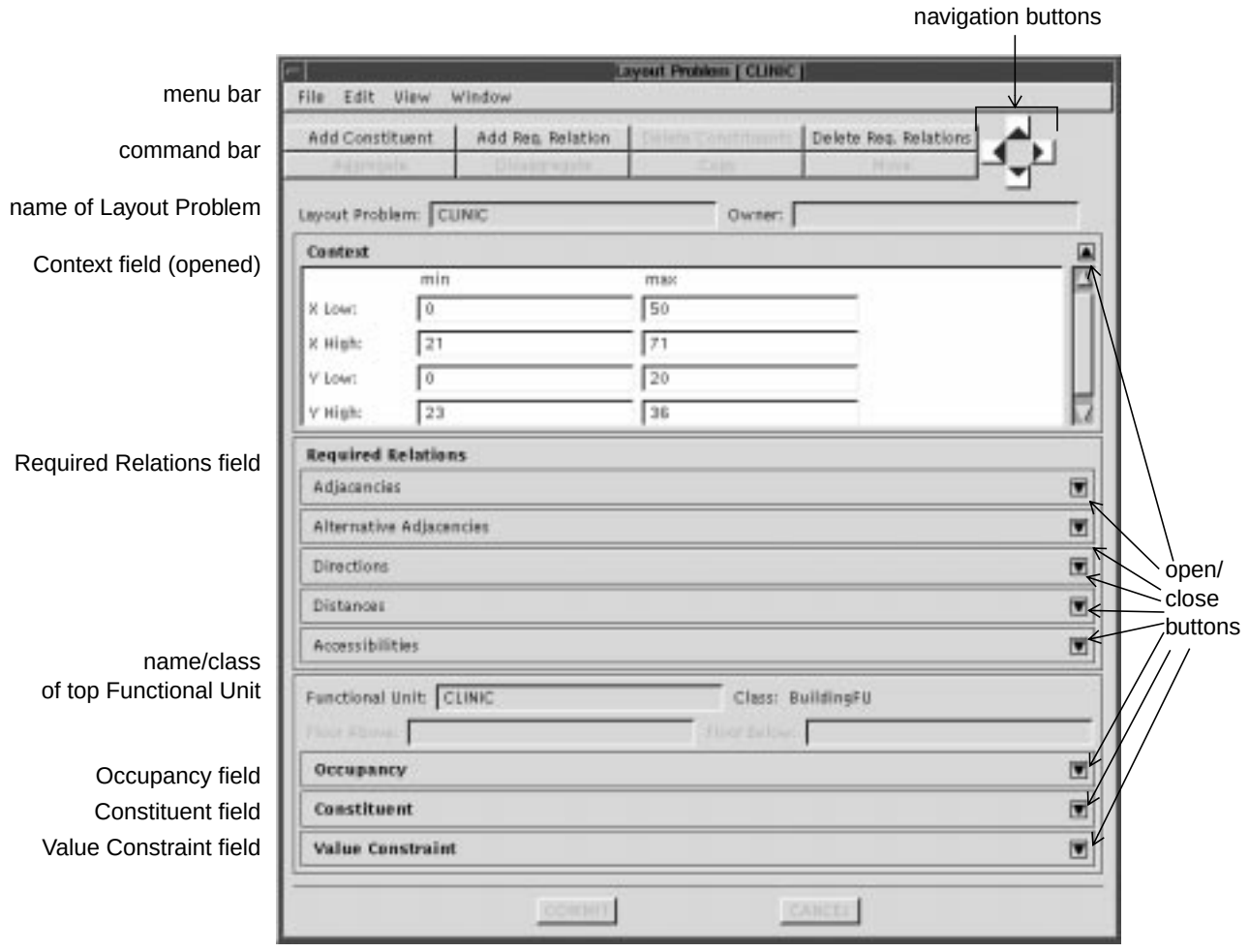


FIGURE 13. Layout Problem Window

5.1 Create an Initial Layout Problem

If you open the Layout Problem Window before any Layout Problem has been opened in the current SEED-Layout session, you may use the commands available from this window to create a new Layout Problem and to activate it. To get started, you must define

- a minimum Context (see Section 5.7 on page 33)
- a top Functional Unit (Section 5.3.1 on page 29, Section 5.3.2 on page 29)
- at least one Constituent for this unit (Section 5.3.3 on page 29, Section 5.6.1 on page 31).

If you then select the Activate Problem command from the Problem menu in the Design Window, SEED-Layout creates an initial Layout that allocates the first Constituent in the defined Context and displays it in the Design Window.

5.2 File Menu

5.2.1 Save Problem

This command saves the active Layout (sub)problem in the file from where it was read.

Caution: The top Functional Unit of the active Layout Problem will be the top Functional Unit overall in the saved problem, and the current Context will be the Context overall independent of where the active problem resides in the current problem hierarchy. That is, if the user is currently working on a subproblem, everything higher up will be lost. To avoid this problem, back up to the highest level Layout Problem before saving it.

5.2.2 Save Problem As ...

This command saves the active Layout Problem as an independent Layout Problem. That is, the top Functional Unit of the active problem will be the top Functional Unit overall in the saved problem, and the current Context will be the Context overall independent of where the active problem resides in the current problem hierarchy.

Steps:

1. **Select the Save Problem As ... option from the File menu. This opens a file dialog box displaying the available problems in the default directory. You may navigate through the directories to open another one.**
2. **Type the name of file in which the problem should be saved in the Save Problem As name field or select the name of the file you want to overwrite in the display field.**
3. **Click Save or Cancel. This closes the file dialog box.**

5.2.3 Close

This option closes the Layout Problem Window.

5.3 Edit Menu

5.3.1 Add Top FU ...

Select this or the following option when you are creating an initial Layout Problem from scratch.

Steps:

1. **Select the Add Top FU ... option in the Edit menu. This opens the Add Top FU dialog box.**
2. **Type-in the name of the top Functional Unit in the Name field.**
3. **Select its class in the Class pop-up menu if you want to override the default selection.**
4. **Select its type in the Type pop-up menu if you want to override the default selection.**
5. **Click COMMIT (or CANCEL to abort the operation).**
6. **Edit the value constraints of the top Functional Unit in the Value Constraint field of the Layout Problem Window. You should set at the least a plausible minimum width and area (see Section 5.9.3 on page 37).**

5.3.2 Add Top FU from Library ...

Select this or the preceding option when you are creating an initial Layout Problem from scratch.

Steps:

1. **Select the Add Top FU from Library ... option in the Edit menu. This opens the FU File dialog box. The box displays instances of all generic Functional Unit classes as well as all other Functional Unit instances that have been saved in the library.**
2. **Select a desired top Functional Unit (see Section 4 on page 25).**
3. **Inspect and, possibly, modify the value constraints of the top Functional Unit in the Value Constraint field of the Layout Problem Window (see Section 5.9.3 on page 37).**

5.3.3 Add Constituent ...

Use this command to add a Constituent to the top Functional Unit. For every Constituent to be added, you must select a class from the following options:

- Functional Unit (default, but not recommended)
- Storey (or Floor)
- Flat roof
- Sloped roof
- Building
- Massing element
- Horizontal zone
- Room
- Shaft
- Atrium

You must also select a type from the following options:

- variable size (default)
- fixed size.

Consult the *SEED-Fundamental Concepts* manual for an explanation of these options.

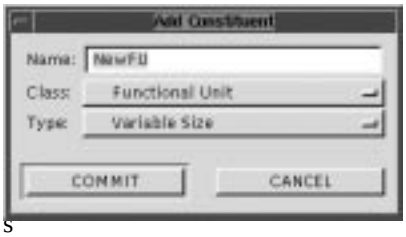


FIGURE 14. Add Constituent Window

s

:

Steps:

1. **Select the Constituent ... option from the Add ... cascading menu in the Edit Menu. This opens the Add Constituent dialog box (see Figure 14).**
2. **Enter the name into the Name field**
3. **Select the class of the Constituent in the Class pop-up menu if you want to override the default selection.**
4. **Select the type in the Type pop-up menu if you want to override the default selection.**
5. **Click COMMIT to confirm the action or CANCEL to cancel.**

5.3.4 Add Constituent from FU Library ...

Select this command to add a Functional Unit that has been saved before as a Constituent of the top Functional Unit.

Steps:

1. **Select the Constituent from FU Library ... option from the Add ... cascading menu in the Edit Menu. This opens the Add Constituent file dialog box that displays the currently saved Functional Units. These include - at the minimum - an instance for each available Functional Unit class.**
2. **Select a desired Functional Unit (see Section 4 on page 25).**
3. **Inspect and, possibly, modify the value constraints of the Functional Unit (see Section 5.9.2 on page 37).**

5.3.5 Share Constituent ...

You may use this command to indicate to SEED-Layout that a Functional Unit that is a VerticalZoneFU (ShaftFU or AtriumFU) should extend into a certain floor. Selecting this command will result in an error message unless each of the following conditions is met:

1. The top Functional Unit of the active Layout Problem is a StoreyFU. That is, you can add a shared constituent only to a StoreyFU and only at the level immediately below.
2. A different StoreyFU constituent in the same MassingElementFU already contains a VerticalZoneFU. Because of 1, this VerticalZoneFU must also be a constituent at the level immediately below that StoreyFU.

Steps:

1. **Select the Share Constituent ... option from the Add ... cascading menu in the Edit Menu. If condition 1 or 2 is not met, SEED-Layout displays an error message and aborts the operation. Otherwise, it opens the Share Constituent dialog box.**
2. **The dialog box displays in its display field the names of the VerticalZoneFUs that can be added as a constituent to the top Functional Unit. Select a name in the field.**
3. **Click Select to complete the operation or Cancel to abort it. If the selected Functional Unit is already a constituent of the top Functional Unit, SEED-Layout displays an error message and abort the operation.**

5.3.6 Add Required Relation ...

This command allows you to add a required relation to the active Layout Problem. Make sure you understand the available relation types and their parameters by consulting the *SEED- Fundamental Concepts* manual if needed.

Steps:

1. **Select the Required Relation ... option from the Add ... cascading menu in the Edit Menu. This opens the Add Required Relation dialog box.**
2. **Select the desired relation type from the available options. If this type differs from the default type, the dialog box may change its appearance to show different parameters.**
3. **Set the parameters in the parameter fields.**
4. **Select the Functional Units to which the relation applies. Except for alternative adjacencies, the relation applies always to two units selected in the FU1 and FU2 fields; you must make four selections for alternative adjacencies to indicate the two pairs of Functional Units involved.**
5. **Click COMMIT (or CANCEL to abort the operation).**

5.4 View Menu

The options in the View menu allow to open/close the following fields of the Layout Problem Window:

- Context
- Occupancy
- Constituents
- Value Constraints
- Adjacencies
- Alt. Adjacencies
- Directions
- Distances
- Accessibilities

5.5 Window Menu

The Window Menu currently contains one option that allows you to open the Constituent Hierarchy Window (see Section 8 on page 43).

5.6 Command Bar

5.6.1 Add Constituent

Use this command to add a Constituent to the top Functional Unit. For every Constituent to be added, you must select a class from the following options:

- Functional Unit (default, but not recommended)
- Storey (or Floor)
- Flat roof
- Sloped roof
- Building

- Massing element
- Horizontal zone
- Room
- Shaft
- Atrium

You must also select a type from the following options:

- variable size (default)
- fixed size.

Consult the *SEED-Fundamental Concepts* manual for an explanation of these options.

Steps:

1. **Select the Constituent ... option from the Add ... cascading menu in the Edit Menu. This opens the Add Constituent dialog box (see Figure 14).**
2. **Enter the name into the Name field**
3. **Select the class of the Constituent in the Class pop-up menu**
4. **Select the type in the Type pop-up menu**
5. **Click COMMIT to confirm the action or CANCEL to cancel.**
6. **(Optional) Inspect and possibly modify the value constraints associated with this new Constituent.**

5.6.2 Add Req. Relation

This command allows you to add a required relation to the active Layout Problem. Make sure you understand the available relation types and their parameters by consulting the *SEED- Fundamental Concepts* manual if needed.

Steps:

1. **Click the Add Required Relation ... button in the command bar of the Layout Problem Window. This opens the Add Required Relation dialog box.**
2. **Select the desired relation type from the available options. If this type differs from the default type, the dialog box may change its appearance to show different parameters.**
3. **Set the parameters in the parameter fields.**
4. **Select the Functional Units to which the relation applies. Except for alternative adjacencies, the relation applies always to two units selected in the FU1 and FU2 fields; you must make four selections for alternative adjacencies to indicate the two pairs of Functional Units involved.**
5. **Click COMMIT (or CANCEL to abort the operation)**

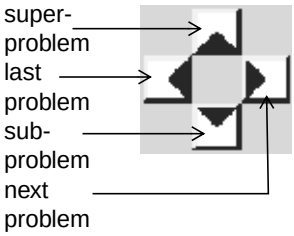
5.6.3 Delete Req. Relations

Steps:

1. **Click the Delete Req. Relations button in the command bar of the Layout Problem Window. This opens the Relational Constraints dialog box.**
2. **(Optional) If the type of relation you want to delete differs from the default type (Adjacency), select the relation type you want to delete in the Type pop-up menu. The display field is updated to display the currently defined relational constraints of the selected type.**
3. **Select the specific constraint you want to delete in the display field. You may make multiple selections by holding down the SHIFT key.**
4. **Click the Remove button.**
5. **(Optional) repeat steps 3 and 4 if you want to delete more constraints.**

6. **Click COMMIT (or CANCEL to abort the operation).**

5.6.4 Navigation Buttons



The navigation buttons at the right end of the command bar offer another means to traverse the problem hierarchy:

Superproblem: Clicking this button re-activates the problem one level up in the hierarchy from the active Layout Problem. SEED-Layout displays an error message if this Layout Problem does not exist. Otherwise, it makes the last active Layout in the associated Design Space the active Layout and displays it in the Design Window.

Last problem: Clicking this button reactivates the last Layout Problem created before the active Layout Problem at the same level in the hierarchy. SEED-Layout displays an error message if this Layout Problem does not exist. Otherwise, it makes the last active Layout in the associated Design Space the active Layout and displays it in the Design Window.

Subproblem: Clicking this button reactivates the first Layout Problem one level down in the hierarchy from the active Layout Problem. SEED-Layout displays an error message if this Layout Problem does not exist. Otherwise, it makes the last active Layout in the associated Design Space the active Layout and displays it in the Design Window.

Next problem: Clicking this button reactivates the first Layout Problem created after the active Layout Problem at the same level in the hierarchy. SEED-Layout displays an error message if this Layout Problem does not exist. Otherwise, it makes the last active Layout in the associated Design Space the active Layout and displays it in the Design Window.

5.7 Context Field

The Context field displays the current Context settings. These settings restrict the overall area available for allocating the top Functional Unit and its Constituents. The meaning of the settings is explained in the table below.

	min	max
X-Low:	lower bound for low x-coordinate of area	upper bound for low x-coordinate of area
X-High:	lower bound for high x-coordinate of area	upper bound for high x-coordinate of area
Y-Low:	lower bound for low y-coordinate of area	upper bound for low y-coordinate of area
Y-High:	lower bound for high y-coordinate of area	upper bound for high y-coordinate of area

You may use the Context field to inspect and modify the Context settings. After a modification, click COMMIT to complete the action or CANCEL to return to the previous settings.

The Context settings are imported from the top Layout Problem if it is opened from a file or computed automatically for subproblems. If you are setting up a new Layout Problem form scratch, you must set at least the max X-High and max Y-High values because otherwise, SEED-Layout does not know where to limit the design area. The min X-Low and Y-low values have zero as default values. Setting a min value equal to a max value fixes the respective coordinate to a single value. You may specify an exact area for allocation by fixing all four coordinates in this fashion.

Note that the settings must be compatible with the value constraints of the top Functional Unit. An incompatible setting occurs, for example, when the minimum width constraint of the top Functional Unit is larger than the difference between the max X_High and min X-Low coordinate: this means that the width of the available area in the x-direction is not large enough to accommodate the required minimum width of the top Functional Unit. SEED-Layout cannot open or activate a Layout Problem with these types of incompatibilities. If you try it, the system responds with an error message.

5.8 Required Relations Field

The subfields of the Required Relations Field allow you to inspect and edit the relational constraints currently set for the active Layout Problem. Note that these constraints are set in the top problem and passed to all subproblems independently of the Constituents of the subproblem. However, when evaluating relational constraints, SEED-Layout currently takes only those constraints into account that apply between Functional Units that are both Constituents in the active Layout Problem. An exception occurs when you request the finalization of the active Layout through all levels. In this case, all relational constraints are instantiated, and SEED-Layout attempts to satisfy all of them.

This approach may appear puzzling at first sight, but makes sense if you think about it. Suppose the Functional Units to which a relational constraint applies are Constituents of different Functional Units, possibly at different levels in the Functional Unit hierarchy. At any time, they may or may not be allocated in a different Design Space, and if they are allocated, they may be allocated in different Layout s. SEED-Layout cannot know against which Layout the constraint should be evaluated; after all, you may revisit that Design Space and generate more alternatives. How could SEED-Layout know what to do in this uncertain situation? It therefore defers consideration of relational constraints between Functional Units at different levels in the hierarchy until you request finalization through all levels (see Section 3.9.9 on page 21). In this case, SEED-Layout knows which Layouts to pick in each Design Space, and it tries to satisfy every relational constraint between Functional Units if each of them has been allocated in an active Layout in some Design Space.

5.8.1 Adjacencies

The Adjacencies field displays the currently valid adjacency constraints. Each row in the field describes a current adjacency constraint. The first two boxes in a row display the names of the Functional Units to which the constraint applies, and the last box defines the minimum overlap required.

Change a setting by overwriting the value in the respective box and by clicking the COMMIT button at the bottom of the Layout Problem Window. Click CANCEL to restore the previous value(s).

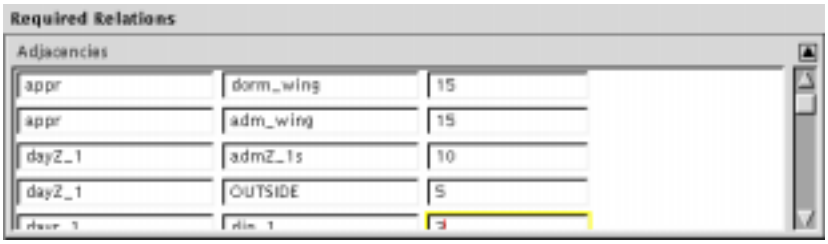


FIGURE 15. Required Adjacencies Field

5.8.2 Alternative Adjacencies

The Alternative Adjacencies field displays the currently valid alternative adjacency constraints. Each constraint is described by two successive rows, each describing one alternative. The first two boxes in a row display the names of the Functional Units to which the alternative constraint applies, and the last box defines the minimum overlap required. Figure 16 displays, for example, two alternative adjacency constraints. The first constraint specifies that the CORRIDOR has to be adjacent either to the WAITING AREA or to the VESTIBULE. SEED-Layout considers an alternative adjacency constraint satisfied if at least one alternative is satisfied.

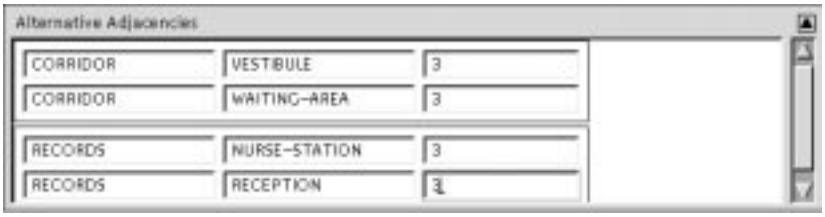


FIGURE 16. Required Alternative Adjacencies Field

Change a setting by overwriting the value in the respective box and by clicking the COMMIT button at the bottom of the Layout Problem Window. Click CANCEL to restore the previous value(s).

5.8.3 Directions

The Directions field displays the currently valid direction constraints. Each constraint is described by a row. The two boxes in a row display the names of the Functional Units to which the constraint applies. The four buttons on the right if the direction indicated by the button is desired. The first constraint in Figure 17, for example, indicates that the appr unit should be to the east or to the west of the dorm_wing unit. SEED-Layout considers a direction constraint satisfied if at least one alternative is satisfied.

Change a setting by overwriting the value in the respective box and by toggling the desired direction button. Click the COMMIT button at the bottom of the Layout Problem Window to confirm or click CANCEL to restore the previous value(s)

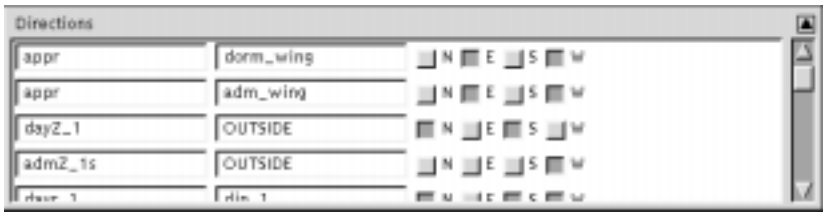


FIGURE 17. Required Directions Field

5.8.4 Distances

The Distances field displays the currently valid distance constraints. Each row in the field describes a current distance constraint. The first two boxes in a row display the names of the Functional Units to which the constraint applies, and the last box defines the maximum distance allowed between the two units.

Change a setting by overwriting the value in the respective box and by clicking the COMMIT button at the bottom of the Layout Problem Window. Click CANCEL to restore the previous value(s).

5.8.5 Accessibilities

The Accessibilities field displays the currently valid accessibility constraints. Each row in the field describes a current accessibility constraint. The first two boxes in a row display the names of the Functional Units to which the constraint applies, and the last box defines the minimum width to be maintained by a public path connecting the two units.

FIGURE 18. Required Accessibilities Field

Change a setting by overwriting the value in the respective box and by clicking the COMMIT button at the bottom of the Layout Problem Window. Click CANCEL to restore the previous value(s).

5.9 Functional Unit Field

The Functional Unit Field shows the name and class of the top Functional Unit of the active Layout Problem and contains expandable subfields displaying its attributes.

5.9.1 Occupancy

The attributes in this field are currently not used in SEED-Layout because they are important only for evaluation software or down-stream processes like structural design that are currently not accessible from SEED-Layout.

5.9.2 Constituents

The Constituents field displays the Constituents of the top Functional Unit. It is these units that are allocated in the Layouts generated for active Layout Problem. Selecting one unit and clicking the right mouse button opens a Functional Unit Window that allows you to inspect and modify the current settings of this Functional Unit (see Section 6 on page 38).

5.9.3 Value Constraint

This field shows the value constraints currently set for the Top Functional Unit. Note that they constrain the overall area available for the Layouts generated for the active Layout Problem.

You may inspect the current settings and edit them. The following table indicates the meaning of each setting. Consult the *SEED - Fundamental Concepts* manual for a more detailed explanation of the settings. Click the COMMIT button when you are done or the CANCEL button to return to the previous settings. See Section 6.6 on page 41 for an explanation of the constraint settings.

6 Functional Unit Window

A Functional Unit Window allows you to inspect and edit the settings of a specific Functional Unit (see Figure 19). It is non-modal, and you may have several open at the same time.

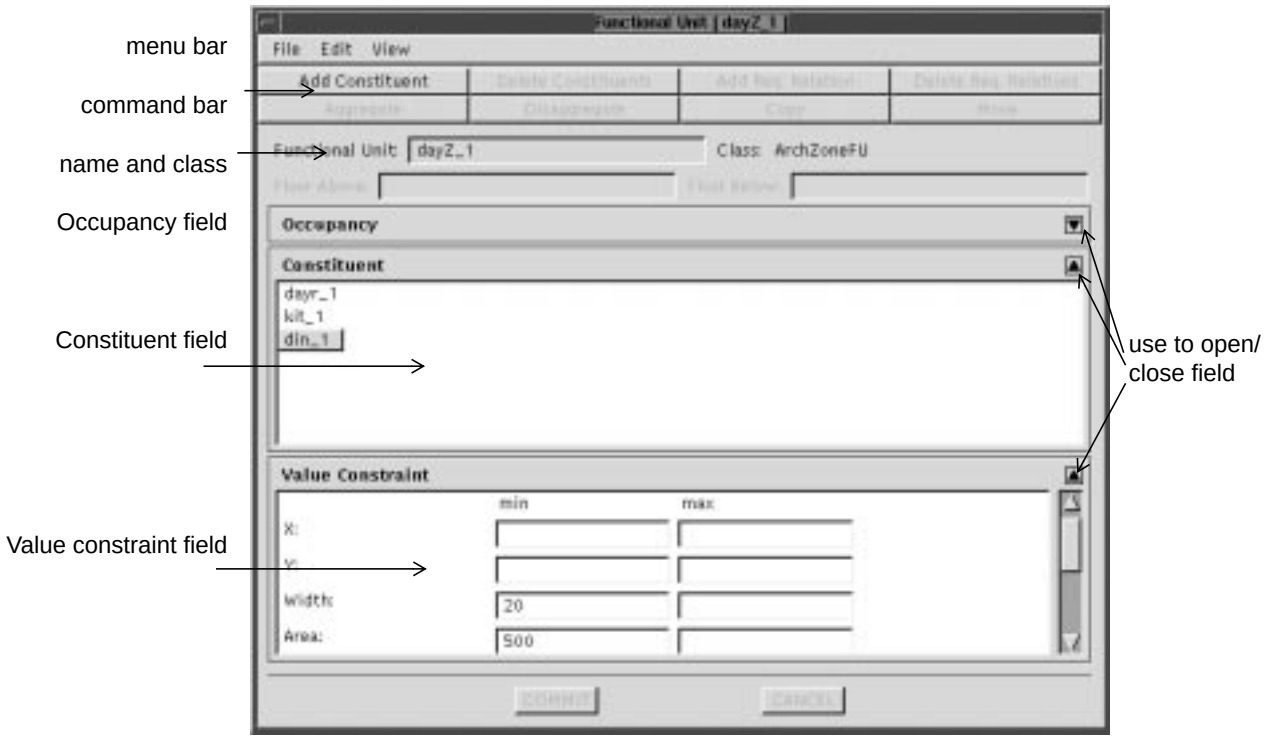


FIGURE 19. Functional Unit Window

6.1 File Menu

6.1.1 Save FU ...

Use this command to save the Functional Unit in the Functional Unit library.

Steps:

1. **Select the Save FU ... option from the Edit menu of the Functional Unit Window. This opens the FU File dialog box.**
2. **Type the name under which you want to save the Functional Unit in the Save Functional Unit field or select the name of an existing Functional Unit in the display field.**
3. **Click Save to complete the action or Cancel to abort it.**

6.1.2 Close

Select this option to close the window.

6.2 Edit Menu

The Edit menu currently contains one pop-up button, Add, from where you may select the following commands.

6.2.1 Add Constituent ...

Use this command to add a Constituent to the Functional Unit. For every Constituent to be added, you must select a class from the following options:

- Functional Unit (default, but not recommended)
- Storey (or Floor)
- Flat roof
- Sloped roof
- Building
- Massing element
- Horizontal zone
- Room
- Shaft
- Atrium

You must also select a type from the following options:

- variable size (default)
- fixed size.

Consult the *SEED-Fundamental Concepts* manual for an explanation of these options.

Steps:

1. **Select the Constituent ... option from the Add ... cascading menu in the Edit Menu. This opens the Add Constituent dialog box (see Figure 14).**
2. **Enter the name into the Name field.**
3. **Select the class of the Constituent in the Class pop-up menu.**
4. **Select the type in the Type pop-up menu.**
5. **Click COMMIT to confirm the action or CANCEL to cancel.**
6. **(Optional) Inspect and possibly modify the value constraints associated with this new Constituent (see Section 6.5 on page 41).**

6.2.2 Add Constituent from FU Library ...

Select this command to add a Constituent to the Functional Unit that has been saved before.

Steps:

1. **Select the Constituent from FU Library ... option from the Add ... cascading menu in the Edit Menu. This opens the Add Constituent file dialog box that displays the currently saved Functional Units. These include - at the minimum - an instance for each available Functional Unit class.**
2. **Select a desired Functional Unit (see Section 4 on page 25).**
3. **(Optional) Inspect and possibly modify the value constraints associated with this new Constituent (see Section 6.5 on page 41).**

Inspect and, possibly, modify the value constraints of the Functional Unit (see Section 5.9.2 on page 37).

6.2.3 Share Constituent ...

You may use this command to indicate to SEED-Layout that a Functional Unit that is a VerticalZoneFU (ShaftFU or AtriumFU) should extend into a certain floor. Selecting this command will result in an error message unless each of the following conditions is met:

1. The Functional Unit displayed in the Functional Unit Window is a StoreyFU. That is, you can add a shared constituent only to a StoreyFU and only at the level immediately below.
2. A different StoreyFU constituent in the same MassingElementFU already contains a VerticalZoneFU. Because of 1, this VerticalZoneFU must also be a constituent at the level immediately below that StoreyFU.

Steps:

1. **Select the Share Constituent ... option from the Add ... cascading menu in the Edit Menu. If condition 1 or 2 is not met, SEED-Layout displays an error message and aborts the operation. Otherwise, it opens the Share Constituent dialog box.**
2. **The dialog box displays in its display field the names of the VerticalZoneFUs that can be added as a constituent to the top Functional Unit. Select a name in the field.**
3. **Click Select to complete the operation or Cancel to abort it. If the selected Functional Unit is already a constituent of the Functional Unit, SEED-Layout displays an error message and abort the operation.**

6.3 View Menu

The options in this menu allow you to open the three attribute fields, which you can also open/close using the arrow buttons at the right-hand end of the respective title bars.

6.3.1 Occupancy

This option opens the Occupancy field. The attributes in this field are currently not used in SEED-Layout because they are important only for evaluation software or down-stream processes like structural design.

6.3.2 Constituents

This option opens the Constituents field.

6.3.3 Value Constraint

This option opens the value constraints field.

6.4 Occupancy Field

The attributes in this field are currently not used in SEED-Layout because they are important only for evaluation software or down-stream processes like structural design that are currently not accessible from SEED-Layout.

6.5 Constituents Field

The Constituents field displays the Constituents of the Functional Unit. Selecting one unit and clicking the right mouse button opens another Functional Unit Window that allows you to inspect and modify the current settings of this Constituent.

6.6 Value Constraints Field

You may inspect the current settings and edit them. The following table indicates the meaning of each setting. Consult the *SEED - Fundamental Concepts* manual for a more detailed explanation of the settings. Click the COMMIT button when you are done or the CANCEL button to return to the previous settings.

	min	max
X:	lower bound for x-coordinate	upper bound for x-coordinate
Y:	lower bound for y-coordinate	upper bound for y-coordinate
Width:	minimum width of unit	maximum width of unit
Area:	minimum area of unit	maximum area of unit
Aspect:		0 - no aspect ratio defined positive value - aspect ratio to be maintained
Height:	minimum floor to ceiling height of unit	maximum floor-to-ceiling height of unit
Elevation:	defined only for storeys (floors): elevation above project zero elevation	
Floor Depth Above	defined only for storeys (floors): assumed depth of floor slab above	

The following table shows the value constraints handled by SEED-Layout for variable-size Functional Units.

The next table shows the constraints handled by SEED-Layout for fixed-size Functional Units.

Long Side:

Short Side:

Height:

Access side: Short side Long side Arbitrary

Elevation:

7 Layout Problem Manager

The Layout Problem Manager is a window that you can open under the Window menu in the Design Window. Figure 20 shows an example.

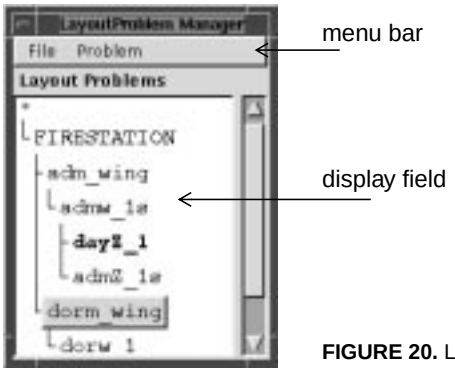


FIGURE 20. Layout Problem Manager

7.1 File Menu

7.1.1 Refresh

<to be added>

7.1.2 Close

This option closes the Layout Problem Manager.

7.2 Problem Menu

7.2.1 Activate

Selecting this option activates the Layout Problem selected in the display field. SEED-Layout updates the Design Window to display the last active Layout in the associated Design Space.

7.2.2 Remove

The remove option is currently not implemented.

7.3 Display Field

This field displays all Layout Problems that have been active so far in the current Design Space. You may select a Layout Problem to highlight it. Clicking the right mouse button opens a pop-up menu that allows you to activate the problem. The remove option is currently not implemented.

8 Constituent Hierarchy Window

Figure 21 shows a snapshot of the Constituent Hierarchy window. You can open this window by selecting the Constituent Hierarchy option under the Window menu in the Design Window. The window displays the Constituents of the top Functional Unit in the active Layout Problem through all lower levels. Each Constituent is shown as the node of a tree whose links indicate the Constituent relations that define the hierarchy.

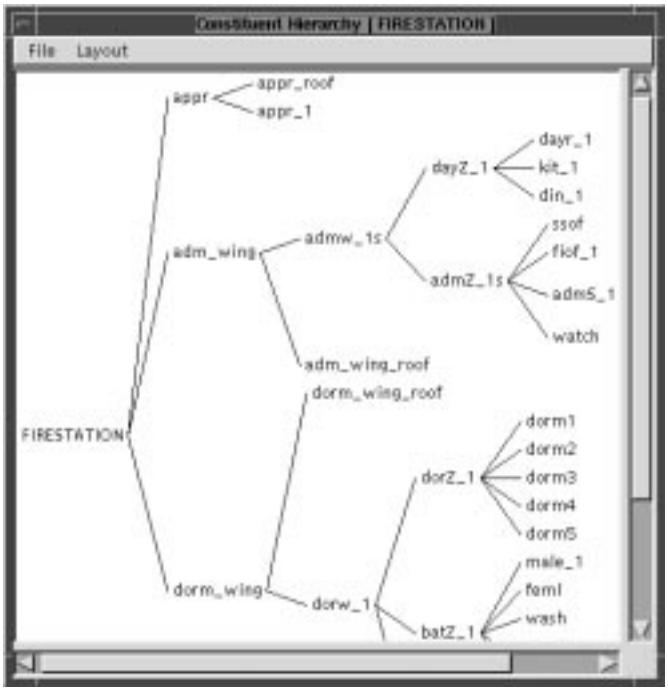


FIGURE 21. Constituent Hierarchy Window

8.1 Edit a Functional Unit

Selecting a Constituent in the display area of the Constituent Hierarchy Window and clicking the right mouse button opens a pop-up menu whose options allow you to edit this Functional Unit.

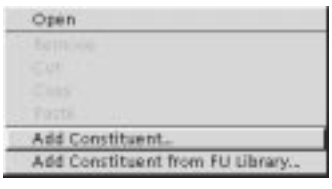


FIGURE 22. Edit Functional Unit menu

The following three options are currently implemented.

8.1.1 Open

This option opens the Functional Unit Window, which you can use to inspect and modify the current settings (see Section 6 on page 38).

8.1.2 Add Constituent

This option opens the Add Constituent Window, which you can use to add a Constituent to the selected Functional Unit.

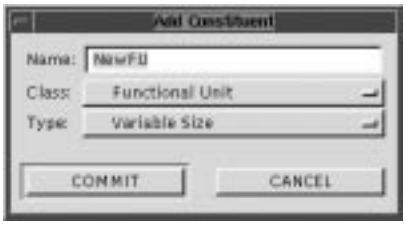


FIGURE 23. Add Constituent Window

For every Functional Unit added, you must select a class from the following options:

- Functional Unit (default, but not recommended)
- Storey (or Floor)
- Flat roof
- Sloped roof
- Building
- Massing element
- Horizontal zone
- Room
- Shaft
- Atrium

You must also select a type from the following options:

- variable size (default)
- fixed size.

Consult the *SEED-Fundamental Concepts* manual for an explanation of these options.

Steps:

1. **Enter the name into the Name field.**
2. **Select the class of the Constituent in the Class pop-up menu.**
3. **Select the type in the Type pop-up menu.**
4. **Click Commit to confirm the action or Cancel to cancel.**

8.1.3 Add Constituent from FU Library

Steps:

1. **Selecting this option opens the Add Constituent file dialog box that displays the currently saved Functional Units. These include - at the minimum - an instance for each available Functional Unit class.**
2. **Select a desired Functional Unit (see Section 4 on page 25).**

(Optional) Inspect and possibly modify the value constraints associated with this new Constituent (see Section 6.5 on page 41)..

8.2 File Menu

8.2.1 Refresh

<to be added>

8.2.2 Close

This option closes the Constituent Hierarchy Window.

8.3 Layout Menu

This menu allows you to change the display of the Constituent hierarchy.

8.3.1 LeftRight

The tree is shown with its nodes arranged from left to right.

8.3.2 TopDown

The tree is shown with its nodes arranged from top to button (as in Figure 21).

8.3.3 Indented

The tree is shown with its nodes arranged from top to button with indentations indicating the different levels.

9 Layout Tree Window

Figure 24 shows a snapshot of the Layout Tree Window. It displays all Layouts that have been generated for the active Layout Problem. The window represents each Layout by a node and shows explicitly the parent/child relations between Layout s. The currently active Layout is indicated by a tick box.

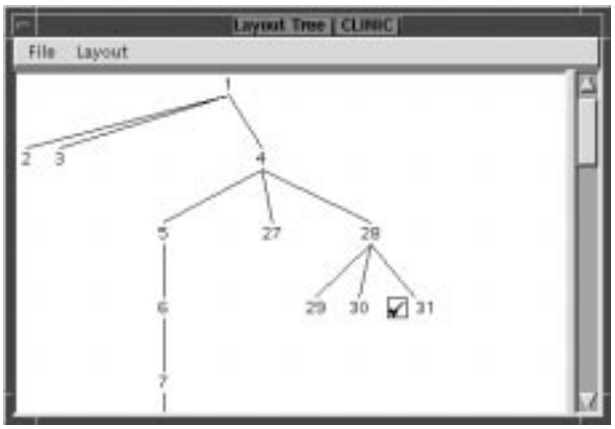


FIGURE 24. Layout Tree Window

You may open the Layout Tree Window by selecting the Layout Tree option under the Window menu in the Design Window.

9.1 File Menu

9.1.1 Refresh

<to be added>

9.1.2 Close

This option closes the Layout Tree Window.

9.2 Layout Menu

This menu allows you to change the display of the Layout tree.

9.2.1 LeftRight

The tree is shown with its nodes arranged from left to right (as in Figure 24).

9.2.2 TopDown

The tree is shown with its nodes arranged from top to button.

9.2.3 Indented

The tree is shown with its nodes arranged from top to button with indentations indicating the different levels.

10 Evaluation Result Window

The Evaluation Result Window reports the results of evaluating the active Layout against the constraints associated with the Functional Units allocated in it. Use the scroll bar to navigate through the report (see Figure 25).

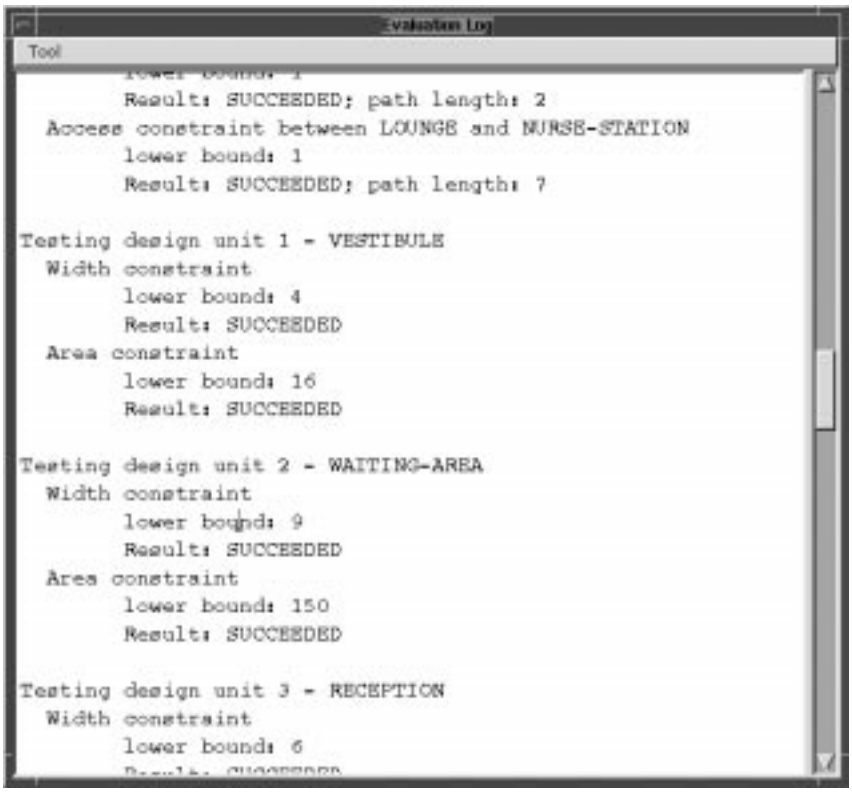


FIGURE 25. Evaluation Result Window

Note that the evaluation results of successive active Layouts accumulate in the window. If you want to reduce its size, select the Clear option in the Tool menu. The Close option in the Tool menu closes the window.

Appendix A

This appendix describes the format for files containing the specification of a Layout Problem that can be loaded into SEED-Layout. Any program that is able to generate a file of this form may serve as front-end to SEED-Layout. The file parser itself has been generated from a high-level grammar specification using the UNIX programming tools lex and yacc[1].

The file format is defined below in terms of a context-free grammar in BNF notation[2]. The only terminal symbols in the grammar are keywords, which are uniformly made up of upper case alphabetic characters and, possibly, underscores. Non-terminal symbols of the grammar are indicated by lower case characters and, possibly, the underscore character and angular brackets. Non-terminal symbols that are enclosed in brackets expand into sequences of symbols. Non-terminal symbols not enclosed in brackets represent character sequences that will be interpreted as single strings or numbers.

A Layout Problem specification is recursively defined as a list of “speclines” of arbitrary length:

```
<specification> → <specline> | <specification> <specline>
```

A `<specline>` must belong to one of the following six types:

```
<specline> → <context_spec>
   | <units_spec>
   | <fu_spec>
   | <valconstraint_spec>
   | <relconstraint_spec>
   | <constituent_spec>
```

A context specification, `<context_spec>`, restricts the overall area available for laying out the Functional Units in the Layout Problem. It may take one of the following forms:

```
<context_spec> → CONTEXT X_LO_MIN number
   | CONTEXT X_LO_MAX number
   | CONTEXT X_HI_MIN number
   | CONTEXT X_HI_MAX number
   | CONTEXT Y_LO_MIN number
   | CONTEXT Y_LO_MAX number
   | CONTEXT Y_HI_MIN number
   | CONTEXT Y_HI_MAX number
```

The keyword `CONTEXT` indicates that the specification under consideration is a context specification. The next keyword indicates which context variable is being specified. `number` must be a real or integer number.

Example: `CONTEXT X_HI_MAX 180.00`

---

1. J. R. Levine; T. Mason; D. Brown: *lex & yacc* (2nd ed.) O’Reilly &Ass., Sebastopol, CA (1992)
2. A. V. Aho; R. Sethi; J. D. Ullman: *Compilers.* Addison-Wesley, Reading, MA (1988) ch. 2

The example context specification restricts the overall extent in the x-direction to the coordinate 180. Note that the parser does not check for consistency between different context specifications. SEED-Layout's constraint solver does this when it tries to generate an initial layout for the Layout Problem defined in an open file.

A unit specification identifies the units of measurement used in the present Layout Problem. It has the following form:

```
<units_spec> → UNITS unit
```

where `unit` must be one of the following one- or two-character strings: `mm, cm, m, in, ft`. The units have the obvious meaning.

Example: `UNITS ft`

A Functional Unit specification,`<fu_spec>`, defines a Functional Unit and can have one of the following forms:

```
<fu_spec> → FU fu_id fu_class fu_name
   | TOP_FU fu_id fu_class fu_name

fu_class →  BUILDING
   | MASS_ELEMENT
   | SHAFT
   | ATRIUM
   | FLOOR
   | FLAT_ROOF
   | SLOPED_ROOF
   | HORIZ_ZONE
   | ROOM
   | FUNCTIONAL_UNIT
```

The first specline must be used to define Functional Unit constituents and the second to define the top Functional Unit of the Layout Problem.

Example: `TOP_FU ae2e88 BUILDING "Maintenance facility"`

`fu_id` is a unique string consisting of alphanumeric characters only. When SEED-Layout generates a Layout Problem file, it uses the pointer values of the respective Functional Unit objects as identifiers, which are hexadecimal numbers. But the parser reads these identifiers as strings so that any other alphanumeric string will do as long as it is unique. Whenever the file parser encounters such an `fu_id` in a specline, it compares it with the entries in a symbol table; if the entry exists, the parser attaches the specification to the corresponding Functional Unit; otherwise, it creates a new entry and associates with it a new Functional Unit. In this way, multiple references to the same Functional Unit in different speclines can be properly interpreted.

`fu_class` is a character string indicating the class of the Functional Unit. Use of the `FUNCTIONAL_UNIT` class is discouraged because this is essentially an abstract class.

`fu_name` is the name of the Functional Unit as it appears in the display of floor plans. It can have one of two forms:

- a string consisting solely of alphanumeric characters and, possibly, the underscore character
- a string enclosed in double quotes, where any character can appear between the quotes except for the double quote itself and the newline character.

A value constraint specification, `<valconstraint_spec>`, specifies a value constraint for the Functional Unit indicated by `fu_id`. It has one of the following forms:

```
<valconstraint_spec> → MIN_WIDTH fu_id number
   | MAX_WIDTH fu_id number
   | MIN_AREA fu_id number
   | MAX_AREA fu_id number
   | ASPECT fu_id number
   | SHORT_SIDE fu_id number
   | LONG_SIDE fu_id number
   | FRONT_LONG fu_id number
   | X_LOW fu_id number
   | Y_LOW fu_id number
   | X_HIGH fu_id number
   | Y_HIGH fu_id number
   | HEIGHT fu_id number
   | ELEVATION fu_id number
```

The example specline below defines a min. area constraint of 1500 (in the units specified by a units specification) for the Functional Unit with id `a34928.`

Example: `MIN_AREA a34928 1500`

In general, the `MIN_AREA, MAX_AREA, MIN_WIDTH, MAX_WIDTH` and `ASPECT` keywords specify value constraints for variable-size Functional Units. The `SHORT_SIDE, LONG_SIDE` and `FRONT_LONG` keywords specify value constraints for fixed-size Functional Units. A `HEIGHT` specification can be given for any Functional Unit. The `ELEVATION` specification is processed only for buildings and floors.

The parser checks only if the numerical constraint parameter in a value constraint specline is in the allowable range; that is, `number` must be always positive, except for `ELEVATION`, where it can also be zero or negative (depending on where the project origin has been placed), and for `FRONT_LONG`, where it must be the integer 0 or 1. The parser does not check for consistency between individual value constraints; it leaves this to the constraint solver during allocation.

A relational constraint specification, `<relconstraint_spec>`, can have one of the following forms:

```
<relconstraint_spec> → ADJACENCY fu-ref fu_ref number
   | ACCESS fu_ref fu_ref number
   | DISTANCE fu_ref fu_ref number
   | DIRECTION fu_ref fu_ref number number number number
   | ALTERNATIVE ad_id fu_ref fu_ref number

fu_ref → fu_id
   |   OUTSIDE
```

The example below specifies that the Functional Unit with id `a2d8c8` must be adjacent to the outside with a minimum overlap of 12 units:

Example: `ADJACENCY a2d8c8 OUTSIDE 12`

If both `fu_ref`s are `fu_id`s, the respective relation is established between the corresponding Functional Units. If one `fu_ref` is the string OUTSIDE in a relational constraint specification, the other `fu_ref` must be an `fu_id` for obvious reasons.

The parser checks if the `number` parameters are positive except for the `DIRECTION` constraint, in which case each `number` must be a binary integer; the four numbers indicate if a direction is desired (1) or not (0) in the following order: N, E, S, W.

The `ALTERNATIVE` specline can be used to specify an individual alternative in an alternative adjacency constraint. `ad_id` is a unique indentifier of the alternative adjacency constraint. It has the same form as and functions like an `fu_id` in that it defines an entry in a symbol table and is used to link individual alternatives to the proper constraint: When the parser encounters an `ad_id` in an `ALTERNATIVE` specline, it checks if that id exists in the symbol table; if so, it links the alternative to the constraint with that id; otherwise, it creates a new alternative adjacency constraint with that id and enters it into the symbol table before linking the alternative to it.

A constituent specification has the following form:

`<constituent_spec>` → `CONSTITUENT fu_id fu_id`

The first `fu_id` identifies the Functional Unit that has the Functional Unit with the second `fu_id` as a constituent. At the end of the parse, SEED-Layout checks if every Functional Units that has been defined is in fact a constituents of another Functionmal Units, except for the top unit. If it finds Functional Units that do not meet this condition, it flags them as 'orphan' units.

Lines that start with the '#' character are treated as comment lines until the newline character is encountered. The first line of the file must be a comment line starting with the string

`#SLayout_Problem`

The file extension must be '.lp'. The Load command uses this extension and the initial line to determine if the file selected by the user is of the proper form. After this opening line, comment lines can appear anywhere in the file and are ignored by the parser.

A specline does not have to end with the newline character. However, it is good practice to do this. It increases the readability of the file, and - more importantly - it provides convenient resynchronization points for the parser when it tries to recover after a syntax error has been encountered; in this case, the parser ignores the current specline that cannot be properly parsed, prints an error message, and restarts parsing after the next newline character. The information in the corrupted specline will not be processed; that is, the corresponding specification will not be set in the Layout Problem.

The order of the speclines in a Layout Problem file is almost arbitrary. For example, you may assign a constituent relation between Functional Units before either one has been defined by an `FU` or `TOP_FU` specline, that is, before the class of either Functional Unit or its name are known. In this case, the parser creates temporary Functional Units with the

proper identifiers and adds specifications to each until it encounters a definition specline that indicates the correct class and name, at which time it replaces the temporary Functional Unit with the desired one under the same id; it also transfers all value constraints and constituent relations that have been collected so far for the temporary Functional Unit to this new version.

There is one exception to this: the ELEVATION constraint cannot be set for a FLOOR or BUILDING before that FLOOR or BUILDING has been defined by an FU or TOP_FU specline. The reason is that FLOORs and BUILDINGs are the only Functional Units for which the ELEVATION constraint can be set. All other Functional Units receive their elevation in a layout from the FLOOR or BUILDING to which they belong. We decided to limit the ELEVATION constraint in this way to make it easier for SEED-Layout to maintain consistency between the elevations of all pieces in a layout, not only when a constituent hierarchy is set up, but also when costituents are moved between floors. Once pitch constraints are implemented for roofs, a similar restriction will hold for the definition of roofs and pitch constraints. In general, it is good practice to define a Functional Unit before any of its attributes is set.

Since the parser is able to recover from errors, it usually creates a Layout Problem that is largely correct. SEED-Layout subsequently attempts to create a design space and an initial layout in that space. It fails to do this only if the following essential parts of a Layout Problem have not been given:

- a top Functional Unit
- at least one constituent of the top Functional Unit
- the X_HI_MAX and Y_HI_MAX context specifications.

SEED-Layout needs these pieces of information because it has to know the maximum extent of the available area and what to allocate in it. Figure 26 shows portions of a complete Layout Problem file that illustrate aspects of the present grammar.

```
#SLayout_Problem
CONTEXT X_LO_MIN 0.00
CONTEXT X_HI_MAX 180.00
CONTEXT Y_LO_MIN 0.00
CONTEXT Y_HI_MAX 50.00
UNITS ft
TOP_FU 72d928 BUILDING Firestation
MIN_WIDTH 72d928 50.00
MIN_AREA 72d928 4400.00
HEIGHT 72d928 15.00
FU 72dab8 MASS_ELEMENT appr
SHORT_SIDE 72dab8 40.00
LONG_SIDE 72dab8 50.00
FRONT_LONG 72dab8 0
HEIGHT 72dab8 15.00
CONSTITUENT 72d928 72dab8
FU 72dbb8 SLOPED_ROOF appr_roof
ELEVATION 72dbb8 15.00
...
CONSTITUENT 72dab8 72dbb8
FU 72de78 FLOOR appr_floor
ELEVATION 72de78 0.00
...
CONSTITUENT 72dab8 72de78
FUu 72e1b8 MASS_ELEMENT “Adm. Wing”
...
Y_LOW 72e1b8 10.00
...
FU 72e2f0 FLOOR Admw_1f
...
CONSTITUENT 72e1b8 72e2f0
...
ADJACENCY 72dab8 72e1b8 12.00
DIRECTION 72dab8 72e1b8 0 1 0 1
...
```

FIGURE 26. Sample Layout Problem file

Index

C

Constituent Hierarchy Window 43
- change tree layout 45
- close 45
- file menu 45
- layout menu 45
- open 10, 32

Context
- settings 34

D

Design Unit
- add 11, 17
- aggregate 14, 19
- change function 8, 18
- disaggregate 15, 20
- edit 8
- expand 13, 20
- remove 12, 18

Design Window 5
- command bar 17
- evaluation menu 17
- File menu 6
- generation buttons 22
- generation menu 11
- Layout menu 8
- layout navigation buttons 23
- problem menu 14
- Unplaced Units dialog box 23
- view control buttons 22
- view menu 9
- window menu 10

E

Evaluation Result Window 48
- open 11

F

File
- open 26
- save 26

File Dialog Box 25
- directories menu 25
- file menu 25
- options menu 25

Functional Unit
- add Constituent 39, 44
- add Constituent from FU library 39, 44
- add shared Constituent 40
- classes 39
- edit 43
- fixed size 41
- occupancy 37, 40
- save in FU library 38
- type 39
- variable size 41
- view Constituents 37, 41
- view value Constraints 41
- view value constraints 37

Functional Unit Window 38
close 38
edit menu 39
file menu 38
open 37, 44
view menu 40

L

Layout
delete 13, 21
finalize 21
finalize+save 7
generate all children 22
generate child 12, 22
generate sibling 12, 22
goto child layout 13, 23
goto last sibling 13, 23
goto next sibling 13, 23
goto parent 13, 23
open 7
save 7
units 9
Layout Problem
activate 42
activate new 14
add Constituent 29, 32
add Constituent from FU library 31
add required relation 31, 33
add shared Constituent 31
add top FU 29
add top FU from lbrary 29
create initial 28
create new 14, 28
delete required relation 33
goto last problem 16, 33
goto next problem 16, 34
goto subproblem 16, 34
goto superproblem 33
remove 42
save 6, 28
save as 28
save document file 16
top Functional Unit 37
view hierarchy 42
Layout Problem Manager 42
close 42
file menu 42
open 11
problem menu 42
Layout Problem Window 27
close 28
command bar 32
Context field 34
edit menu 29
file menu 28
Functional Unit field 37
navigation buttons 33
open 10
required relations field 34
view menu 32
Layout Tree Window 46
change tree layout 46
close 46

file menu 46
layout menu 46
open 10

M

MDS file
output unit 9
save 7
snap grid 9

R

Required relations
accessibilities 37
adjacencies 35
alternative adjacencies 35
directions 36
distances 36
instantiation 35

S

SEED-Layout
quit 8
Semper 17

U

Unplaced Unit
change allocation order 23

V

View
access paths 9
actual size 10
adjust scale 22
sublayout 13
superlayout 9, 13
zoom in 22
zoom out 22

Z

Zoom
in 10
out 10