SEED-Layout Tutorial

Ulrich Flemming

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

Jan. 31, 1999

1. Introduction

1.1 Overview

Welcome to the tutorial for a software application called SEED-Layout. This software was developed within the School of Architecture at Carnegie Mellon University. Its intended area of application is the schematic phase of layout design, which determines the overall organization and configuration of a planned building.

In this chapter we will explain the basic task addressed by SEED-Layout. We will use the following tutorial chapters to introduce the basic concepts underlying SEED-Layout and its capabilities in a structured, step-by-step fashion. The tutorial stars with very basic concepts and operations and progresses to concepts and operations of increasing complexity. The chapters build upon each other so that later chapters will be difficult to understand if the material presented in earlier chapters has not been mastered.

We assume in this tutorial that the user has installed the latest working version of SEED-Layout on a Windows 2000 or Windows XP computer. In order to gain the most from the tutorial, the user is expected to complete the exercises as described. Several of these exercises rely on data files residing under the "data" folder or directory.

1.2 Purpose of tutorial

This tutorial has the following goals:

- Introduce the schematic layout software tool called SEED-Layout and explain some of its conceptual underpinnings
- Provide examples of how SEED-Layout is intended to be used, showing inputs t o and outputs from the tool
- Introduce the SEED-Layout user interface and how users can interact with it to create schematic layouts. This is done by providing a series of exercises at increasing levels of difficulty and complexity.

1.3 Intended tutorial audience

The following tutorial is meant to suit the following audiences:

- Those who may need to use SEED-Layout to perform a practical architectural schematic layout task.
- Those who are presently engaged in preliminary building design and wish to explore how SEED-Layout might assist this process
- Those who wish to familiarize themselves with the generative approach to design support, of which the SEED tools are good examples.
- Those who wish to study also the capabilities of SEED-Pro, the architectural programming module of SEED. One purpose of SEED-Pro is to generate input for SEED-Layout in the form of an architectural program that forms the basis for the schematic layout task. The concepts underlying SEED-Pro are much better understood if users know how SEED-Layout will eventually make use of them.

1.4 How to approach this tutorial

The following approach is recommended:

- Test whether your PC copy of SEED-Layout can be launched properly. If not, reinstall the application and pay attention to the proper setting of the SEED environment variables, without which SEED-Layout will not launch successfully on a PC.
- Finish reading the introduction to this tutorial manual in order to get a better understanding of what ‘schematic layout design’ means in the context of SEED-Layout. Make sure that the SEED-Layout Reference Manual is in easy reach.
- Complete the SEED-Layout tutorial chapters in the given order including all exercises. It is advised that you do this over more than one day because some concepts used in SEED-Layout are novel and not part of traditional architectural practice. They were introduced in order to take full advantage of the computer’s power, but it may take some time until they fully ‘sink in’.
- Progress to the SEED-Pro tutorial if you so wish.

1.5 Text formatting conventions used in tutorials

In the exercises, we observe the following text formatting conventions:

- Windows and menu items to be selected or opened in SEED-Layout are abbreviated as follows:
  From Menu: **Specifications > Edit Buildings > New Building...**
  This means the following:
  From the '**Specifications'** menu, select the **'Edit Buildings'** sub-menu, then select the '**New Building...'** command. These types of abbreviations are common in software tutorials and reference manuals and should be easy to follow.
- Words intended to be entered by the user into a SEED-Layout dialog box are shown in a fixed-width font: e.g. enter: `Clinic.`
- Definitions of technical terms or concepts that require a more detailed explanation are shown as follows:

SEED-Layout An application which assists the schematic layout phase in building design.

1.6 What is SEED-Layout?

SEED-Layout is one of the software applications comprising the SEED environment. It specifically assists architects in the schematic layout phase in which they derive the overall building organization and configuration (which will eventually include aspects of site design). SEED-Layout is meant to support an exploratory mode of design by encouraging architects to derive and compare conceptual alternatives before they zero-in on a specific design concept.

1.7 Task

Schematic layout design is an integral component of the early building design phase; we outline in the present section briefly how this task is understood in the SEED context. The succeeding chapters introduce you to the specific concepts SEED-Layout uses to support this task and to the operations it offers the user to create schematic layouts rapidly.

Figure 1 shows examples of schematic layouts as the term is understood in the context of SEED-Layout. Schematic layouts are typically drawn with 'center-lines', which delineate the important functional areas and major circulation elements on a floor. In the finished design, physical spatial enclosures and partitions may be placed on (parts of) the center lines and will then have a positive thickness. But a center line drawing abstracts from such details. It is typically drawn at 1/16" or, metrically, 1/200 scale; this scale makes it difficult to represent physical building elements, while helping designers understand how the overall building is organized and how it fits into the site and surrounding context.

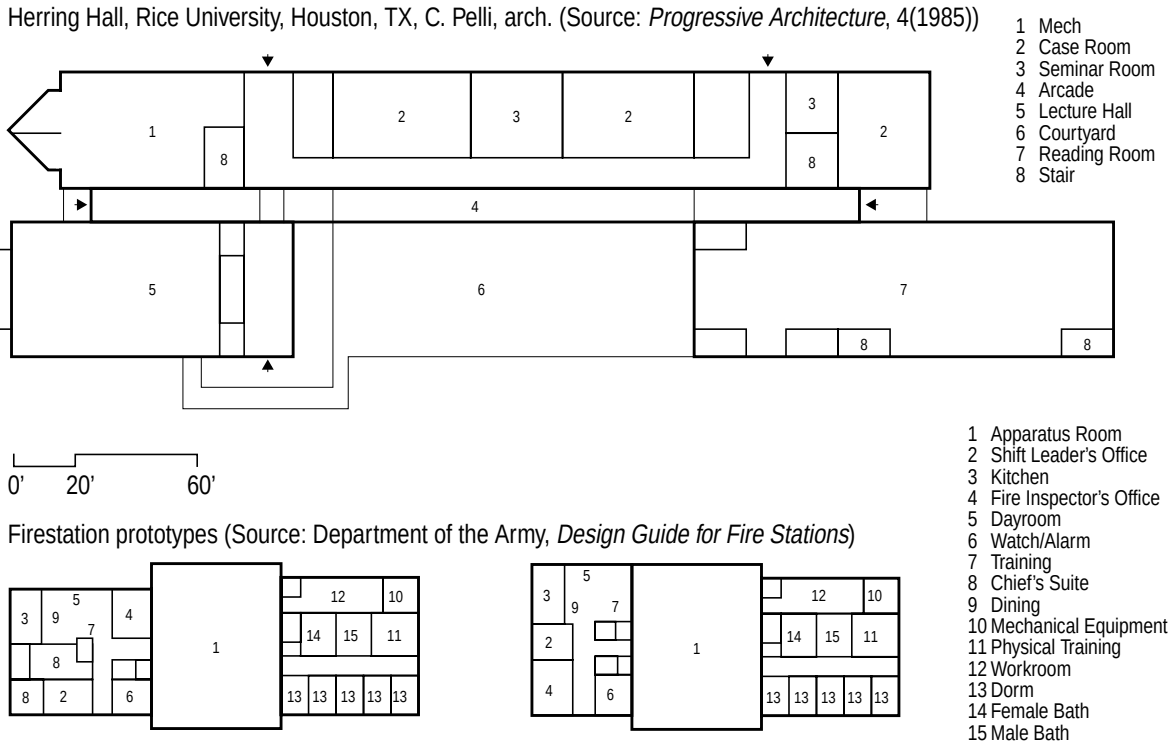


FIGURE 1. Examples of schematic layouts

The schematic layouts of the various floors of a building indicate its overall massing, which influences - among other aspects - its performance with respect to energy consumption or costs. They determine also basic circulation or movement patterns, both on a floor and between floors. Aside from such functional and practical aspects, these layouts determine the overall form of the building and how it fits into the given context; they are thus an important indicator of the architect's formal or expressive intent.

Especially larger buildings are often divided into distinct parts. The firestations in Figure 1 consist, for example, of a central apparatus room and two wings. The wings are lower than the central part, require an independent roof, and have their own foundation. This organization is a standard scheme developed by the US Army Corps of Engineers, one of the main sponsors of SEED, for firestations built on Army bases.

The second example in Figure 1, the first floor of Herring Hall at Rice University, illustrates another important organizational strategy found specifically in larger institutional buildings, zoning. Herring Hall is clearly organized into parallel zones, which determine basic circulation patterns; they also influence the structural grid and thermal zones that the structural and mechanical engineers, respectively, will use.

The zones clearly exert a strong influence on the overall massing and form of the building: they are articulated as distinct parts through independent roofs, variations in materials etc. and can be ‘read’ as such from the outside. Zoning is thus an organizational strategy likely to have far-reaching implications for the form and function of a building.

SEED-Layout supports schematic layout design at the level of detail found in center line drawings. It supports specifically the use of basic organizational devices like wings or zones and is meant to encourage designers to experiment with alternative organizational strategies. To achieve this goal, it provides designers with a range of generation and evaluation capabilities that can be activated with ease and in any combination.

1.8 Preview

Session 1: SEED-Layout Fundamentals
Basic concepts and commands introduced in the context small single-storey buildings.

Session 2: Hierarchical Layout Generation
SEED-Layout’s way to handle multi-storey buildings, zones and other building subdivisions.

Session 3: The SEED-Layout Design Space
The most complex notion used by SEED-Layout is that of a design space, introduced in this section. If you master this section, you’re home free.

2. *Session 1: SEED-Layout Fundamentals*

Prerequisites: none

Approximate time to complete:

2.1 Overview

This chapter concentrates on the following topics:

- SEED-Layout Problem and its parts: how does SEED-Layout know what to design?
- Schematic layouts and their generation: what are the parts of a schematic layout in SEED-Layout and how can they be generated or placed?

In dealing with these issues, SEED-Layout relies on a few basic concepts: Layout Problem, Context, Functional Units, Design Units and Walls, which we will introduce in this chapter.

We introduce these concepts in the context a very simple layout task: to generate the layout of a small clinic for ambulatory health care administered by a single physician or by two physicians with alternating schedules (see Figure 2).

FIGURE 2. Target Layout CLINIC

2.2 Getting Started

1. Open the directory where the SEED-Layout executable is located (typically C:\seed)
2. Double-click on the SEED-Layout icon; after a short wait, the SEED-Layout application opens.

Note:
If SEED-Layout does not open at this point, it is likely that the required environment variables have not been set properly in your computer.

The readme.txt file provided as part of the SEED-Layout package contains instructions on how to set these variables.

2.3 The SEED-Layout 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.

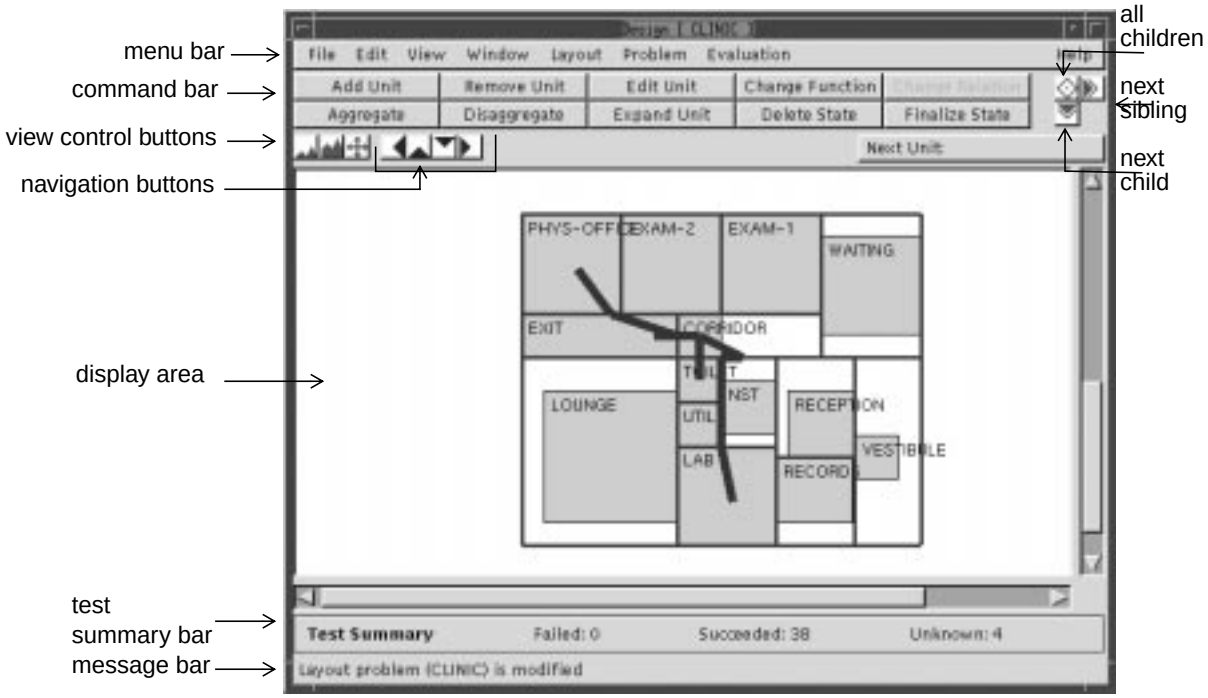


FIGURE 3. Design Window Components

Figure 3 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 users 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 users 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.

Load the Layout Problem that contains the spatial program for the clinic:

Steps:

1. File > Load Problem... opens a file dialog box displaying the available Layout Problems in the default directory or a directory structure. Navigate through data->PS_LIB until you find the Layout Problem file **`CLINIC-TUTORIAL.lp`**.
2. Click the Open button. SEED-Layout loads the CLINIC Layout Problem into memory and displays an initial Layout.

We will inspect the parts of a Layout Problem in the next exercises, using the CLINIC as an example.

2.4 Functional Units - Design Units

Inspect the Layout in the Design Window. Why does it show a rectangle labelled VEST in the middle of a larger area?

In order to find an answer to that question, open another window called the Layout Problem Window:

Window > Layout Problem

Open the Constituent field by clicking the arrow button at the right end of the title bar. You notice that it shows a list of names that designate the functional areas that comprise the clinic we want to layout. The unit VEST, which stands for vestibule, is at the top of the list; that’s why it shows up in the initial Layout. Note also that the next unit is called WAIT.

Refocus the Design Window without closing the Layout Problem Window (you can always reopen it if you closed it inadvertently). Click the **Next Child** navigation button at the lower right corner of the command bar. The layout is redrawn showing now VEST

and WAIT, with WAIT placed below VEST. You may guess that WAIT was the next unit placed because it followed VEST in the Constituent list in the Layout Problem Window.

But why are the units placed in the middle of the area shown? Refocus the Layout Problem Window and click on VEST in the Constituent field; the name will be highlighted. Click the right mouse button. This brings up a dialog box; select the View Constituent option. This opens a Functional Unit Window. Inspect the Value Constraint field by opening it as you opened the Constituent filed in the Layout Problem Window; notice that it contains the following settings:

Min. width = 4; min. area = 16

Close the Functional Unit window for VEST and open the Functional Unit Window for WAIT. Going through the same steps, you'll notice

Min. width = 9; min. area = 150.

The values that we found are spatial requirements associated with VEST and WAIT, respectively. SEED-Layout tries to satisfy these requirements automatically whenever this is possible. How does it do this?

Refocus the Design Window. Select VEST in the layout by clicking it with the left mouse button. Click the Edit button in the command bar. This brings up a dialog box showing the current settings of the VEST area in the layout. You'll notice that it shows 8 values (see Figure 4). These values indicate the lower and upper bounds within which the coordinates of VEST can vary. For example, the X_Low coordinate, which is the x-coordinate of the left corners, can vary between 0 and 67; the other coordinates are similarly defined.

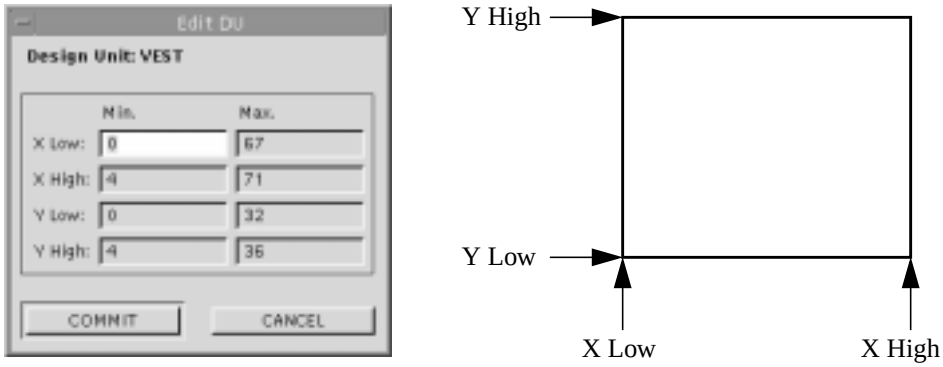


FIGURE 4. Dimensional range available for VEST

Notice that the min X High value is 4 ft. greater than the X Low value: this is a direct consequence of the min. width requirement of 4 feet; that is to say, SEED-Layout

deduces from the requirement that the left x coordinate has to be at least 4 feet higher than the low x-coordinate. Try to understand the other coordinate values in the same way.

But notice also that the coordinates themselves have a great degree of variability. This is due to the fact that not much has been placed in the area that is overall available. This variability is SEED-Layout's way of indicating that VEST is placed 'somewhere' in the plan. Before dismissing the Edit window, note that the maximum Y High coordinate for VEST is 21.

Dismiss the Edit window by clicking the CANCEL button. Select WAIT in the Layout and open its Edit window. Try to understand its settings in terms of the requirements specified for WAIT. Notice also that its max y-coordinate is 32. This explains why the min y-coordinate of VEST is only 9: since WAIT has to be at least 9 ft. wide and since it is placed below VEST, VEST cannot get closer than 9 feet to the lower boundary of the layout. (We'll see soon how we know that this is the lower boundary).

We are now ready to introduce some fundamental concepts of SEED-Layout. You have noticed that there are two different elements that have the name VEST, one is a rectangle in a layout, and one is a constituent in a layout problem. What's the difference, and why do they have the same name?

Let's start with the area labelled VEST in the layout. This is an instance of what is called a ***Design Unit***. This name has to be understood in the broader context of SEED:

Design Unit

A Design Unit (DU for short) is a building component. It may be spatial, like a vestibule or an office zone, or physical, like a window or a floor slab. Design Units are the primary focus of design in SEED. SEED-Layout deals only with spatial Design Units.

The constituent called VEST is called a Functional Unit. Again, Functional Units are concepts shared between the individual modules of SEED:

Functional Unit

A Functional Unit (FU for short) collects all requirements that apply to a specific Design Unit. SEED-Layout uses these requirements when it has to place, size or resize a Design Unit.

Why separate Design and Functional Units? There are many reasons for this separation, which we will discuss during the exercises. For starters, observe that when we generate layout alternatives, we can have Design Units in different layouts that have the same associated Functional Unit. For example, we may have one alternative in which the vestibule is in the lower right corner, while in another plan, it's in the center. Separating the reqirements from the area to which they apply allows us to express the

requirements only once and in one place. This also allows us to edit the requirements once and then propagate them to all associated Design Units. One of the most basic features you have to understand when working with SEED-Layout is the distinction between Functional and Design Units.

When a Design Unit is placed to satisfy the requirements of a specific Functional Unit, we say that it ***allocates*** that Functional Unit. When drawing a layout in the Design Window, SEED-Layout simply labels each Design Unit with the name of the Functional Unit it allocates.

Exercises:

1. Inspect the requirements of some other constituents in the current Layout Problem
2. Without looking back, try to express in your own words what Design and Functional Units are and why we distinguish between them.

2.5 Layout Problems

One question remained from the preceding exercises: How does SEED-Layout know that 36 ft. is the maximal y-coordinate in the overall available area? Refocus the Layout Problem Window, close the Constituent Field and open the Context field. You'll find eight attributes that look very similar to the coordinates that indicate the locational possibilities for a Design Unit. They have indeed a similar function: they indicate the variability available in the overall area.

Observe that the maximum value for the high y-coordinate is 36 ft. That's how SEED-Layout gets this value. More explanations...

We can now express succinctly what constitutes a Layout Problem in SEED-Layout:

Layout Problem A Layout Problem consists of a Context and a list of constituent Functional Units with associated requirements.

The task designers can set for themselves when working with SEED-Layout is to generate layouts that allocate the Functional Units in the given Context so that as many requirements as possible are satisfied. Designers are specifically encouraged to explore alternative possibilities.

But what about requirements that deal with relations between Functional Units? They can also be part of a Layout Problem. Inspect the Required Relations field in the Layout Problem window. You'll notice five bars labelled Adjacencies, Alternative Adjacencies, Directions, Distances, Accessibilities, which indicate the five types of required relations SEED-Layout is currently able to handle.

Click the arrow button on the right-hand side of the Adjacencies bar. This opens a scrollable window listing the required adjacency relations for our CLINIC. The first two fields in each relatuioin indicate the Functional Units between which the relationship must be maintained. The third field contains a number. A required adjacency is interpreted very strictly by SEED-Layout: the Design Units allocating the respective Functional Units must have a shared boundary; the number in the third field indicates how long this boundary has to be at the minimum (see Figure 5). This will guarantee that whatever traffic is expected to pass between the two Design Unit will find an opening that is sufficiently large. Consult the Reference Manual, section 5.8, if you want to learn about the other types of required relations handled by SEED-Layout.

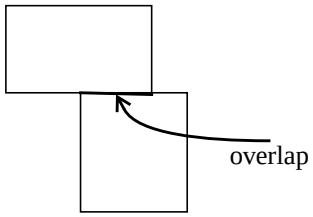


FIGURE 5. Two adjacent Design Units

SEED-Layout lets you know about how well a layout that you generated satisfies the requirements: after each step, it displays an evaluation summary at the bottom of the design window. If you want to see more details, open the Evaluation Results Window:

Window > Evaluation Results

This window keeps a running commentary; if scrolling gets too cumbersome, eliminate past reports by selecting the Clear option in the Tool menu.

Exercise:
Try to understand how the report for the last layout generated corresponds to the requirements in the CLINIC problem (Hint: reopen the appropriate windows from the Layout Problem Window).

2.6 Walls

Click the Next sibling button in the Design Window. Another layout appears in this window in which WAIT is to the left of VEST. Clicking again produces a layout in which WAIT is above VEST. If you look at the target layout, you'll see that this is where it should actually be. Can we generate a desired placement quicker and more directly? Yes, but before we exercise the commands that allow us to do that, we should have a closer look at the lines SEED-Layout draws between Design Units. These lines are called Walls.

Walls

In a final layout, walls indicate lines on which physical walls can be placed, and that's how they got their name. But in intermediate layouts in which Design Units have no fixed locations, they play a somewhat different role: they indicate basic spatial relations between Design Units that SEED-Layout will maintain in a specific layout.

The last three layouts we generated are different in that WAIT and VEST have different relations, which are indicated by the walls separating the units from each other.

You may also look at walls as the indicators of the basic zones you create on a floor: the wall that cuts across the last layout establishes two parallel horizontal zones. WAIT and VEST are allowed to glide along the zone divider or to move to the right or left as more units are added; but they will never cross the wall (unless you remove and reinsert one of the units). At this point, you have the option to continue these horizontal zones through the rest of the layout; to restrict this zoning to only a part of the plan and introduce another scheme in other portions; or to pursue a different zoning idea, in which case you will have go back to a different layout or generate one that has not been produced so far.

If we now look at the target layout, we see that the horizontal zones established in the last layout are supposed to continue through its entire length. Knowing this, we can streamline the process by taking charge of where Design Units are going to be placed.

What is the next Functional Unit scheduled for allocation? Look at the Next Unit button below the right end of the command bar: it displays the name of the unit that will be placed next. In this particular case, it's the CORRIDOR. If you look at the target layout, the CORRIDOR is supposed to be in the upper zone, i. e. above VEST (albeit not necessarily adjacent to it) and to the left of WAIT. We can tell SEED-Layout that it should place the CORRIDOR in these relations to the existing units by selecting WAIT and the wall to the left of WAIT:

1. Click WAIT; it will be highlighted
2. Hold down SHIFT and click the left wall; it will be highlighted, too (you may reverse steps 1 and 2!)
3. Click the Add Unit button in the command bar.

SEED-Layout interprets this command with the present selections as follows: it allocates CORRIDOR by 'pushing' WAIT away from the selected wall. The new layout in the Design Window shows the result.

Let's figure out how to place EXAM-1 in the desired position. It's supposed to be to the left of WAIT and above CORRIDOR. Select WAIT and again its left wall and click Add. What happened? WAIT is pushed away from the left wall all right and EXAM-1 is inserted in the gap, but it's placed between WAIT and the CORRIDOR, not above the CORRIDOR.

Let's try another option. Click the Goto Parent navigation button. This returns us to the previous layout. Select CORRIDOR and the above wall and click Add: this pushes CORRIDOR down from the above wall and places EXAM-1 above the CORRIDOR; as a side effect, it places EXAM-1 in the desired relation with WAIT.

Observe that we have now established another horizontal zone above the corridor that is supposed to span between WAIT and the left boundary of the layout.

Exercise:
Try to allocate EXAM-2 and PHYS (the physician's office) in that zone. If you don't get it right the first time, keep on returning to the parent layout(s) until you succeed.

2.7 The Layout Tree

You may have guessed that it's possible to revisit not only the parent layout of a layout, but also other layouts that you generated before. This is indeed the case, and SEED-Layout provides two basic devices; the navigation buttons, of which the Goto Parent button is a par, also contain a First Child, Last sibling and Next Sibling button. Try these for the currently active Layout. We call this local navigation because you move only one step away at a time from the currently active layout.

Try to return to the layout that was active at the beginning of this section. If you have trouble finding it again, open the Layout Tree Window:

Window > Layout Tree.

This window shows little boxes that represent all the layouts generated so far, each with a link to its parent. The currently active layout is marked. You can use this tree not only to get an overview of what you have done so far, but also to reactivate directly any layout in the tree: to do this,

1. select the respective box
2. press the right mouse button and select activate in the dialog box that comes up

We call this ***global navigation*** because you can make jumps of arbitrary distance.

Exercises:
1. Experiment with the different options offered in the Layout menu of the Layout Tree Window (sorry for this re-use of the term "layout", for which we are not responsible: it's part of the application framework in which SEED-Layout is written. Without this framework, we couldn't be able to change the "layout" of trees).
2. Reactivate the last layout we generated and insert REC (reception) left of VEST.

2.8 Layout Generation - Review

We have worked so far with the following layout generation commands:

- Add Design Unit: This command allows you to place a Functional Unit in desired relations with existing units. SEED-Layout always tries to complete this command, even if the specified location results in some constraint violations. If constraints are violated, an error message will be displayed.
- Next Child: SEED-Layout generates the next child layout from a parent layout. If no child has been generated so far, the next child will actually be the first child.
- Next Sibling: SEED-Layout generates the next sibling of the active layout, which is the same as the next child of the parent layout.

You may construct a complete layout step-by-step using these commands, where the Functional Units are allocated in the given order. If this were all you had at your disposal, layout design in SEED-Layout would be a lock-step process with little flexibility. The following sections introduce commands that add flexibility to the process.

But before we introduce these possibilities, a few additional remarks about the last two commands, which we call “semi-automated” layout commands, appear in order. These commands may not succeed. This happens when the requirements that are applicable based on the Functional Units involved contradict each other or the constraints implied by the geometry of the active layout. In this case, no first or next child or no next sibling that satisfies all applicable requirements exists: SEED-Layout gives up and aborts the operation. That is to say, it does not try to find something like the “next best” solution. The reason is that finding this solution in the face of multiple and possibly multiply conflicting requirements is computationally non-trivial. We have made no attempt to solve this problem.

You must keep this in mind especially if you generate with the Add command layouts that contain constraint violations: you cannot generate a next child automatically from this parent because the constraint violations will be inherited by all children (except for very rare cases involving pin-wheel configurations that we will introduce below). Aside from backing up to a different branch of the Layout tree, you have various options, which we will introduce later on.

2.9 Changing the Allocation Order

Let’s start with the order of allocation. The layout that we have produced so far should look like Figure 6

We have completed the upper zone and want to start developing the lower zone. The next unit scheduled for allocation is N-ST, the nurses’ station. Suppose we want to start

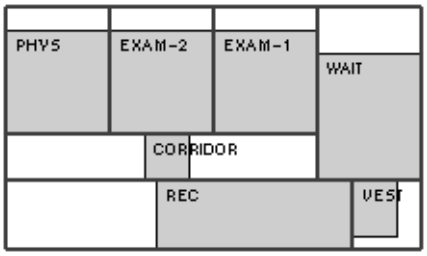


FIGURE 6. Intermediate CLINIC Layout

work on the lower zone by allocating the staff lounge, LOUNGE, next. This requires execution of the following steps:

Moving a Functional Unit to the top of the allocation order:

1. Pull down the Unplaced Units dialog box from the Unplaced Units button. Explain box...
2. Select LOUNGE in the Available field and click the right-pointing arrow button. This moves LOUNGE to the top of the allocation order as indicated in the Selected field.
3. (Optional) you may move other Functional Units up in the order by repeating step 2.
4. Click Commit.
5. Click Dismiss to close the window.

Now allocate LOUNGE as shown in the target layout, that is, to the left of REC and below the CORRIDOR. In case you forgot how to do this: Select REC and the left exterior wall, then click the Add command button.

Exercise:
Select LAB as the next unit to be allocated and locate it in the lower zone to the left of LOUNGE (this does not correspond to the desired location in the target layout, but you'll understand soon why we did this).

2.10 Rearranging a Layout

If you did not complete the last exercise, select LAB now as the next unit to be allocated and try to locate it in the lower zone to the left of LOUNGE (select LOUNGE and the left exterior wall, then click the Add command button).

We now have a situation in which LOUNGE and LAB each occupies a position that the other should have. In order to correct this quickly, execute the following task:

Swap two Functional Units:

1. Select the two Design Units that allocate the Functional Units.

2. Click the Change Function command button. The Layout is updated so that the Functional Units assigned to the selected Design Units are exchanged.

This operation illustrates another advantage of separating Functional and Design Units. Swapping Functional Units simply means swapping the associations with Design Units and recomputing the requirements. If Functional Units and Design Units were inseparable, we could not just swap them. We would have to remove each and reinsert each at a different place or do some other awkward manipulation, which SEED-Layout doesn't even provide for.

Let's now assume for argument's sake that you do not like the location of the physician's office after all and that you want to change it. You may back up in the Layout Tree to the Layout that was active before the PHYS was placed and generate from this layout a new one with PHYS in a different position. But this would be awkward because you would lose the placement of LOUNGE and LAB, which we like. It is better in this situation to remove PHYS from the current layout and re-insert it there:

Remove a Design Unit:

1. Select the Design Unit to be removed, in this case, the one labeled PHYS.
2. Click the Remove Unit command button. The Layout in the Design Window is redrawn to show a layout in which.
3. (Optional) PHYS is placed at the bottom of the allocation order. If you want to re-allocate it right away, move it to the top of the order as explained above.

Exercises:

1. To practice the last two operations, reinsert PHYS between EXAM-1 and EXAM-2 and then swap it with EXAM-2.
2. Allocate the remaining four Functional Units in the given order.

2.11 Finalize

We now have a layout in which all Functional Units in the CLINIC have been allocated so that all requirements are satisfied. But the Design Units have still variable dimensions. The software to which we want to ship a layout for further processing typically expect fixed dimensions. We can create this type of layout by a process called finalize in SEED-Layout:

Finalize

1. Click the Finalize command button. This opens the a dialog box, asking you if you wish to finalize the layout through the entire hierarchy
2. Click the No option (we'll learn in the next chapter what this question really means). The layout is redrawn so that the Design Units fill tightly the enclosing box. If you

inspect the coordinates of a Design Unit (remember how to do this?), you'll see that the min and max values for each coordinate are the same, i. e. they are fixed.

Now suppose that you don't want a CLINIC that forms "boring" box: you want to move the front of the VEST a little back to give the entrance some articulation. At the same time, you may want to indent the left side at the exit.

Manually edit Design Unit coordinates
(not completely implemented!):

1. Select the Design Unit
2. Click the Edit Unit button. This opens the Edit Design Unit window.
3. Type the desired values in the respective field; make sure that min max values remain the same!

Note: You can edit Design Unit coordinates also for layouts that have not been finalized.

We have generated a complete, if simple layout using fundamental commands. In the remainder of this chapter, we introduce additional commands dealing with situations that are less straightforward than the ones we have encountered so far, but typical for practical applications.

2.12 A Closer Look at the Add Design Unit Command

The Add Design Unit command served as the work horse in our work so far. This is usually the case when you have a fairly good idea about the zoning you want to achieve and where things should be placed relative to each other.

Design Units on same side of Wall

In all applications of the command so far, we selected one Design Unit and one Wall. This will not do in all situations. Suppose you do not like the final CLINIC layout we have produced because you think RECORDS should be placed below N-ST and REC(eption) to generate a configuration as shown in Figure 7.

Let's see how we can do this fast. Back up to the parent layout (because it is not yet finalized) and remove RECORDS from it. If you now try to re-insert it by pushing a single unit away from the obvious wall (the lower exterior wall) it so that it ends up in the desired position, you seem to be stuck: no matter which unit you push away from the lower wall, it does not produce the desired result.

What you have to do is push N-ST, REC and UTIL back together to create the space you need:

1. Select the lower wall, N-ST and REC in any order.

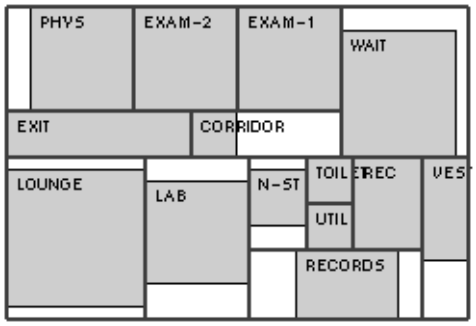


FIGURE 7. New location for RECORDS

2. Click the Add Unit command button. The new layout shows RECORDS in a position created by pushing N-ST, REC and everything in-between away from the wall.

In general then, you can select one or two Design Units and a Wall if you want to execute the Add Unit command. The Design Units must be adjacent to the wall; SEED-Layout checks for this before it executes the command (try to trick it).

Design Units on opposite sides of Wall: Pinwheels

But do the Units have to face the wall from the same side? The answer is no. If they face the wall from opposing sides, SEED-Layout creates a “pin-wheel” or “spiral” configuration. If you want to see an example, do the following:

1. Remove TOILET from the active Layout.

2. In the new Layout, select CORRIDOR, REC and the horizontal wall between them. Click the Add Unit button. You’ll see a pinwheel configuration of walls around the TOILET in the newly created Layout (Figure 8).

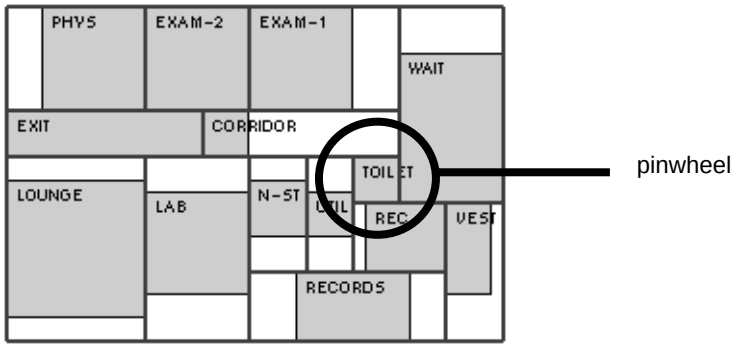


FIGURE 8. Pinwheel configuration of Walls

2.13 Problem Modification

Back up twice to revisit the complete Layout we liked so far. Suppose now that you want to add a short hallway connecting the main corridor with RECORDS, making UTIL accessible at the same time. In order to do this, we have to add a constituent to the CLINIC and then allocate it.

Add constituent
(ancillary hallway)

1. (Re) open the Layout Problem Window.
2. Click the Add Constituent button. This opens the Add Constituent dialog box.
3. Write a name, for example, HALL, in the name field.
4. Select Architectural Zone as type.
5. Click Commit. This closes the dialog box.
6. Open the Constituent field in the Layout Problem window. Select the new constituent and open the Functional Unit window to inspect the default settings. Increase the minimum width to 3 or 4 feet.
7. Click Commit
8. Close the Constituent field.

In all Layouts that we have seen, the EXIT extends toward the right so that PHYS and LOUNGE may be accessible only from the exit. This may be fine if the EXIT is actually part of CORRIDOR. But suppose you want to make sure that the CORRIDOR extends always far enough to the left so that PHYS can be reached from it. One way to do this is to add a required adjacency between PHYS and CORRIDOR.

Add required relation
(adjacency between PHYS and CORRIDOR):

1. Click the Add Required Relation button in the Layout Problem window. This opens the Add Required Relation dialog box.
2. Select the desired relation type (adjacency is the default).
3. Select PHYS and CORRIDOR from the FU1 and FU2 pop-up menus.
4. Set the minimum overlap parameter.
5. Click Commit.

Exercise:
Add the HALL to the Layout as discussed to generate the plan shown in Figure 9.

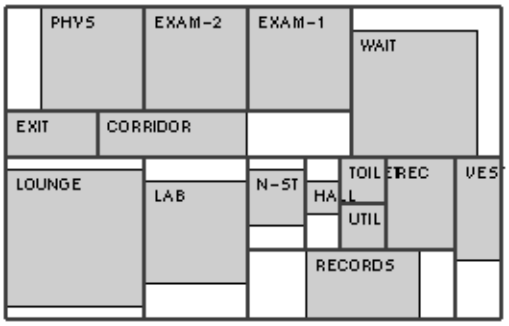


FIGURE 9. CLINIC with ancillary hall

2.14 Extended Exercise

Load the problem RanchFloor, which contains a “bare-bones” program for a raised-ranch suburban residence. Add relational constraints if you like. Use this problem to generate a raised ranch plan, either the simplest solution shown in Figure 10 or the increasingly more elaborate solutions shown in Figure 11 and Figure 12; the difference is that for the latter problems, you will have to extend the Layout Problem as given by adding a second bathroom, hall, even closets. The last plan is interesting because it demonstrates that for tight layouts, pinwheel configurations are indispensable (see the secondary hall in the back of the plan).

FIGURE 10. Minimal raised ranch plan

FIGURE 11. Raised ranch with second bathroom

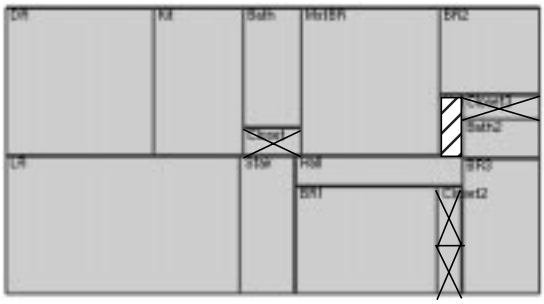


FIGURE 12. Four-bedroom raised ranch

3. *Session 2: Hierarchical Layout Generation*

Prerequisites: successful completion of session 1.

Note: **The HOUSE problem does not work properly in this PC version!**

3.1 Overview

This chapter introduces the major devices SEED-Layouts offers designers who wish to create layouts with specific organizational characteristics. The following concepts and commands are introduced in this chapter:

- Constituent hierarchies
- Problems and subproblems
- Layouts and sublayouts
- SEED-Layout outputs

3.2 Constituent Hierarchies

Load the Layout Problem HOUSE-Tutorial.lp and open the Constituent Hierarchy Window:

Window > Constituent Hierarchy

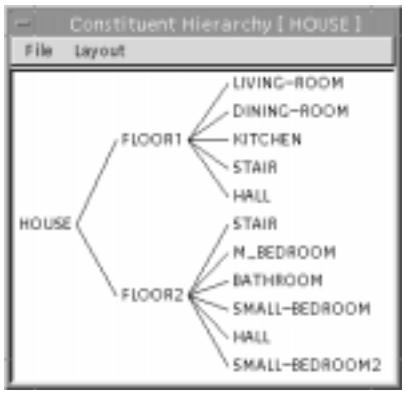


FIGURE 13. Constituent Hierarchy Window for HOUSE

You see a tree diagram depicting Functional Units on two floors. Note that SEED-Layout considers all nodes in the diagram Functional Units: the entire house, its two floors, and the rooms on each floor. We say that the two floors are constituents of the house, and the rooms on each floor are constituents of that floor. Taken together, these constituent relations form what we call a ***constituent hierarchy***.

Constituent Hierarchy A constituent hierarchy consists of Functional Units that contain other Functional Units as constituents. SEED-Layout interprets the constituent relations as spatial containment relations: When a Functional Unit is allocated by a Design Unit, its constituents will be allocated strictly inside this Design Unit.

Requirements are associated with each Functional Unit at each level in a constituent hierarchy. You may inspect these requirements for each Functional Unit from the Constituent Hierarchy Window:

Inspect the requirements of a Functional Unit from the Constituent Hierarchy Window

1. Select the Functional Unit in the Constituent Hierarchy Window;
2. Click the right mouse button; this brings up a dialog box;
3. Select the Open command. This brings up a Functional Unit Window that allows you to inspect the requirements associated with the selected Functional Unit.

Exercise:
Inspect the min. area requirement for a floor and the min. area requirements of its Functional Unit constituents; you will see that the min. area of the floor is at least as large as the sum of the min. area requirements for the individual rooms on the floor.

3.3 Sublayout Problems

Observe that the first constituent allocated in the current Layout is FLOOR1. Click the Next Child button in the Design Window. This allocates the second floor on top of the first floor.

This is to say, SEED-Layout knows that floors cannot be placed next to each other and automatically assumes that they are placed on top of each other. How does it know which floor is on top of which floor?

Open the Functional Unit Window for one of the floors, either from the Constituent Hierarchy Window or from the Layout Problem Window as explained in session 1. You will notice that the Floor Above or Floor Below field indicates which floor is above or below the floor you are looking at.

Note that the Add Design Unit command featured prominently in Session 1 does not work for floors or storeys because they cannot be placed next to each other. To place a floor constituent, you must use the Next Child button in the Design Window.

Let's now try to allocate the Functional Units on the first floor:

Expand a Floor/Storey

1. Move cursor over stacked floors.
2. Click the right mouse button. This brings up a pop-up menu giving you the option of expanding FLOOR-1 or FLOOR-2.
3. Select Expand FLOOR-1. The display in the Design Window changes to show the first floor with its first constituent, LIVING-ROOM, allocated in it.

Selecting a Design Unit in order to allocate constituent Functional Units inside its boundary is called ***expanding*** the Design Unit. It does not matter if we are expanding a Design Unit associated with a floor or any other type of Functional Unit. The only precondition is that the Functional Unit has constituent Functional Units; otherwise, there would be nothing to place. This will become clearer later on in this session.

Inspect the new Layout Problem in the Layout Problem and Constituent Hierarchy Windows. The Layout Problem described in these windows is considered a ***subproblem*** of the overall Layout Problem by SEED-Layout. This subproblem is now the ***active*** Layout Problem.

Look at both Context and Constituents of the active Layout Problem. Observe that this subproblem has the same kinds of parts as the overall problem, a Context and Constituents; this is always true in SEED-Layout: no matter at which level you are, a Layout Problem always has a Context and Constituents.

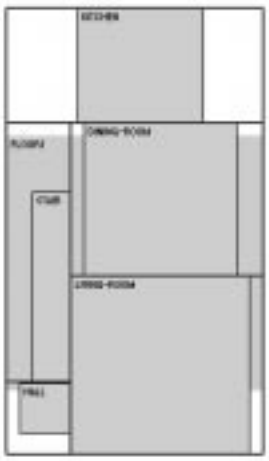
You do not have to specify the Context for subproblems: SEED-Layout finds it automatically by looking at the coordinates of the Design Unit that is being expanded.

Exercise:
Allocate the rest of the Functional Units on the first floor. Try to find a layout that satisfies all requirements. Refer to Figure 14a to see a Layout that will work.

3.4 The Problem Hierarchy

Make sure that the active Layout Problem is the FLOOR-1 subproblem. In the Design Window, the

Problem > Goto Superproblem



a) Floor 1

b) Floor 2

FIGURE 14. House floors, unfinalized

command gets you back to the higher-level problem of which the current Layout Problem is a subproblem. Observe that the Design Window simultaneously displays the last active Layout in the superproblem. This Layout Problem is now the active one and the Layout in the Design Window is now the active Layout. You could continue to work on this Layout if there were more Functional Units at this level to be allocated. In short, the Goto Superproblem command reactivates a Layout Problem and a Layout.

Expand the second floor as you did with the first floor: move the cursor over the stacked floors, press the right button and select Expand FLOOR-2. SEED-Layout creates a new subproblem at the same level as the FLOOR-1 problem. This subproblem is now the active Layout Problem. Try to generate a layout compatible with your first floor layout (see Figure 14b).

Go back to the superproblem; observe how the active Layout and Layout Problem change again. The

Problem > Goto subproblem

command, in turn, (re)activates a subproblem. But note that while before this command activated the FLOOR-1 subproblem, it now activates the FLOOR-2 subproblem. That is to say, the Goto subproblem command always activates the last subproblem of the superproblem that has been active. The

Problem > Goto last problem

command activates a subproblem at the same level in the problem hierarchy that was created before the current one, in our case, the FLOOR-1 problem. If this last subproblem does not exist, the command fails, and SEED-Layout displays an error message.

Problem > Goto next problem

also activates a subproblem at the same level in the problem hierarchy. But it was created after the current one, in our case, the FLOOR-2 problem. Again, this command fails if this next subproblem does not exist.

Problem Hierarchy

The super/subproblem relations between Layout Problems form a Problem Hierarchy. There is always one active Layout Problem. Activating Layout Problems that have been active before is called ***navigating the (Layout) Problem Hierarchy***.

Note that the Goto commands introduced here do not create new Layout Problems. They are strictly navigation commands that allow you to revisit or (re)activate Layout Problems that have been created before. Subproblems can be created only with the Expand command.

Open the Layout Problem Window for the FLOOR-2 subproblem. In the upper right-hand corner are four navigation buttons that allow you to do the same types of navigations that can be done with the Goto commands available in the Problem menu:

Up arrow: activates superproblem

Down arrow: activates last active subproblem

Right arrow: activates next subproblem

Left arrow: activates last subproblem.

Exercise:
Finalize the HOUSE layout. Don't forget to check the yes box when asked if you want finalization through all levels. Navigate again through the Layout Problem hierarchy and inspect the finalized layouts at the super and subproblem levels (see Figure 15).

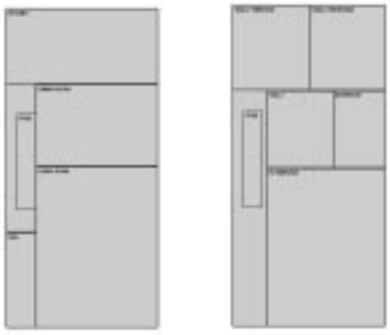


FIGURE 15. Finalized HOUSE floors

3.5 Vertical Zones

Inspect the coordinates of the STAIR on each floor in the finalized HOUSE; you see that they are the same, that is, the STAIR is correctly stacked (see Figure a). How does SEED-Layout know when to do this? Open the Functional Unit Window for STAIR and observe that the STAIR Functional Unit is of type Shaft FU. This is a special version of a vertical zone, which SEED-Layout calls Vert Zone FU. Whenever SEED-Layout encounters a Design Unit associated with a vertical zone during finalization, it checks all floors in order to see if they contain Design Units associated with this Functional Unit and sets the coordinates of the respective Design Units equal to each other. You may check the evaluation window to find the shared wall constraints SEED-Layout sets up to accomplish thse alignments across floors. Figure b illustrates this for the HOUSE problem.

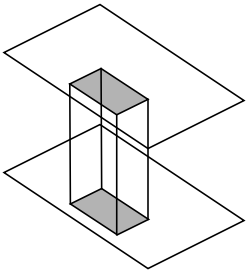


FIGURE 16. Shafts

a. A shaft extending through two floors

b. Shared wall constraints for a shaft on two floors

```
Testing design unit 0 - FLOOR2
  Width constraint
        lower bound: 240
        Result: SUCCEEDED

Testing design unit 1 - STAIR
  Shared north wall with STAIR on floor 1
        Result: SUCCEEDED
  Shared east wall with STAIR on floor 1
        Result: SUCCEEDED
  Shared south wall with STAIR on floor 1
        Result: SUCCEEDED
  Shared west wall with STAIR on floor 1
        Result: SUCCEEDED
  Fixed size constraint
        short side: 36, long side: 180
        Result: SUCCEEDED
```

In order for this to work, you must make sure that when you allocate a vertical zone on different floors, the respective Design Units are stacked roughly on top of each other. Why doesn't SEED-Layout check this by itself? This will become clear only when you understand SEED-Layout's notion of a design space, which is elaborated in the next session. Let's just say for now that for each floor, you are able to generate all kinds of layout alternatives, which may contain the same vertical zone in very different locations. Some alternatives on one floor may be compatible with some alternatives on the other floor, but not with others. How is SEED-Layout to know which alternatives to select on the different floors in checking for compatibility? After all, you may even reactivate a subproblem you worked on before and add yet more layout alternatives.

SEED-Layout checks for vertical compatibility only during finalization when you ask it to finalize across all levels in the hierarchy. It then selects for each subproblem the last active layout and generates shared wall constraints between the Design Units on different floors that are associated with the same vertical zone. SEED-Layout then attempts to finalize each of floor so that, taken together, the floor layouts are compatible.

When finalizing, then, remember this: do not attempt this until you are sure that the last Layout that was active for each subproblem is the one you actually want to be included in the finalization. In ascertaining this, you may want to reactivate some subproblems, look at the active Layouts, and possibly change them if they are not the desired ones.

3.6 More Functional Unit Types

Load now the FIRESTATION problem and inspect its Constituent Hierarchy. If you open the appropriate Functional Unit Windows, you will see that the top Functional Unit is a Building FU. It has three constituents, each of which is a Mass. Element FU, which stands for “massing element”. All massing elements have a ground floor and a roof (a roof is a special kind of floor), represented by Floor or Storey FUs and Roof FUs. Two of the massing elements, the administrative wing and the dormitory wing, have floors that are divided into zones, represented by Horiz. Zone FUs. The zones, in turn, contain Room FUs, if not smaller zones. Figure 13 illustrates this using slightly more comprehensible names (the names in Layout Problem files tend to be short so that they do not clutter Layout displays in the Design Window too much.

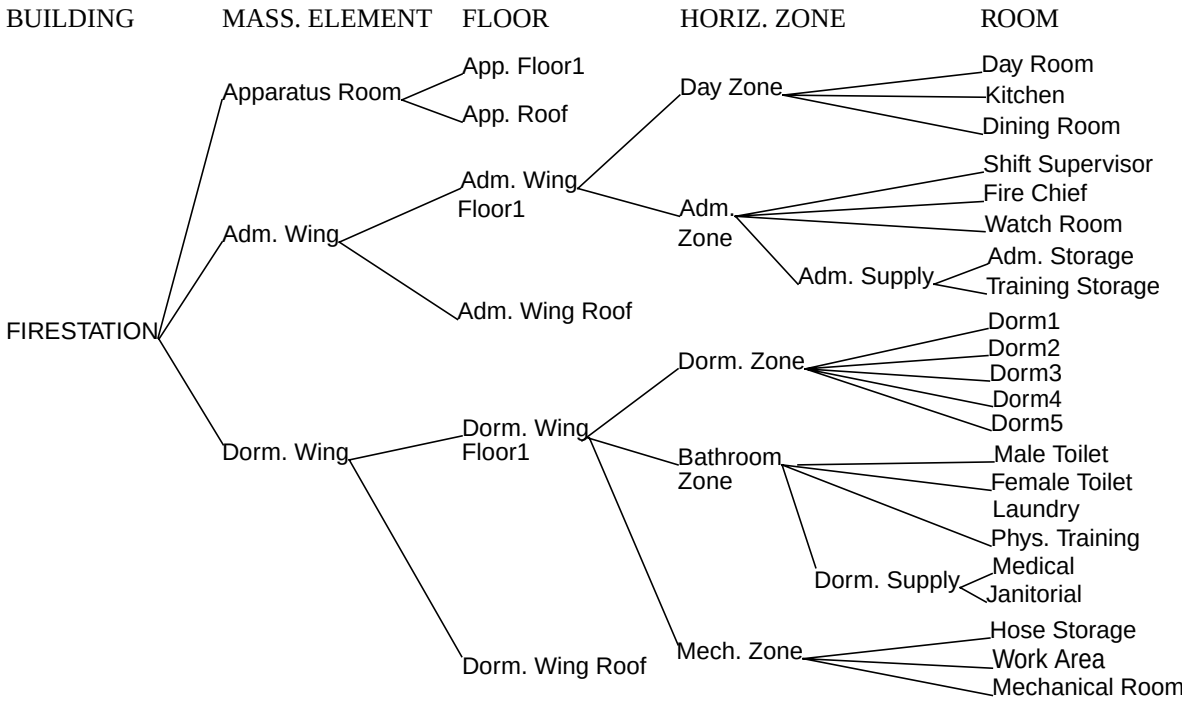


FIGURE 17. The constituent hierarchy for FIRESTATION

The containment relations between these types of Functional Units should become clear once you have gone through the following exercises. For starters, allocate quickly the two FIRESTATION wings to the left and right of the apparatus room to create an overall Layout as shown in Figure 18; you may use the Add Design Unit command introduced in Session 1 or the Next Child/Next sibling buttons until you find the desired configuration. Unlike floors, massing elements clearly behave during allocation like any other Functional Unit.

Let's now layout the administrative wing. Note that this is not a floor stacked on top or below another floor, but an element laid out horizontally with other elements. Expansion is a little different in this case:

Expand a Design Unit that is not a Floor:

1. Select the Design Unit, in our case, adm_wing.
2. Click the Expand button in the command bar of the Design Window. You will see how the display changes to show the first floor of the adm. wing allocated.

Note that like the HOUSE, the constituents of the administrative wing are all floors. Its first floor has been placed. You may click the Next child button to place also the roof. If we want to develop the first floor in the administrative wing further, we face the situation known from the HOUSE: in order to expand a floor, we move the cursor over the stacked floors, press the right mouse button and select the desired expand command, in our case, Expand admw_1s.

In the new Layout Problem, the two Functional Units to be allocated are horizontal zones, and the first of these, the day zone, has been allocated. Allocate the administrative zone admZ_1s below the day zone. You will notice that horizontal zones can be allocated like rooms or massing elements. Note also that by stepping through the subproblems level-by-level and allocating Functional Units at each level, we are realizing a specific organizational scheme that is expressed by the overall constituent hierarchy of FIRESTATION.

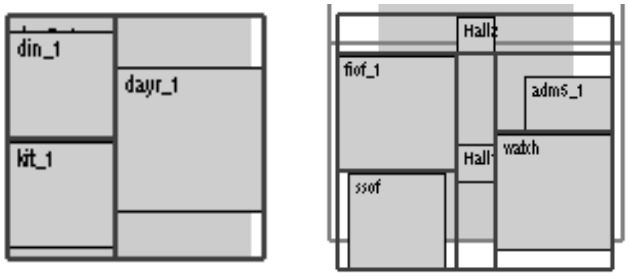
Try now to complete the FIRESTATION layout by doing the following exercises, which require more and more independence and sophistication on your part.

FIGURE 18. Top-level layout of FIRESTATION

Exercises:

1. Expand the day zone to create a layout of its Functional Units as shown in Figure 19.
2. Reactivate the superproblem and expand the administrative zone. Place the four given constituents first, then create two new Functional Units constituents, Hall and Hall2, to create a good circulation system as shown in Figure 19.

3. If you mastered the previous exercises, reactivate the top Layout Problem and expand the dorm wing. In order to accomplish this, you need to apply everything you have learned so far: how to allocate Functional Units including floors, how to expand floors and other types of Functional Units, how to navigate through problem hierarchies, and how to add Functional Unit constituents.
4. When you are done, finalize the entire FIRESTATION.



Day zone Administrative zone

FIGURE 19. Sublayouts in the administrative wing of FIRESTATION

Inspect the finalized layouts at all levels and see how they fit together. SEED-Layout tries to fill holes on floors and to make stacked floors fit into a rectangular box (unless you explicitly modified the floor coordinates of the respective Design Unit or created boundary constraints in the associated Functional Unit to prevent this, for example, to create a terraced mass). But SEED-Layout does not try to press massing elements into an overall box; it actually tries to find the minimal footprint for each massing element (again, you may counteract this by explicitly overwriting the Design Unit coordinates of a massing element or by setting boundary constraints in the associated Functional Unit; see, for example, the boundary constraint for the dorm. wing).

3.7 Functional Unit Types: Summary

For easy reference, we summarize below what you observed or may have guessed about the constituent relations between the different types of Functional Units. The Fundamental Concepts manual gives more details. It specifically shows how very intricate massing/zoning relations can be established with these types of FUs even if they can be allocated only by rectangular Design Units.

Building FU

A building is typically the top Functional Unit in a complex, multi-floor Layout Problem. Its constituents are either all massing elements; all floors (if it is to be considered a single ‘box’); or only room and horizontal zone constituents (if you are not interested in other floors).

Massing Element FU A massing element is a building constituent with no other elements above or below; that is, it is allocated directly on the site. It must be either a top Functional Unit or the constituent of a building; that is, it cannot be contained in any other type of Functional Unit. Its constituents are either all floors or only rooms and horizontal zones. Use massing elements when you knows before-hand that the planned building consists of separate components or wings. A good example is the firestation prototype with its central apparatus room and two wings at opposite ends, each of which constitutes an independent massing element in the SEED-Layout sense.

Floor/storey FU A floor or storey can only be the constituent of a massing element or a building. Different massing elements may have different numbers of floors. Use the elevation attribute to indicate which floors are continuous across massing elements (see Figure 20 for the containment relations between buildings, massing elements and floors). A floor may contain among its constituents only vertical zones like shafts, horizontal zones, or rooms.

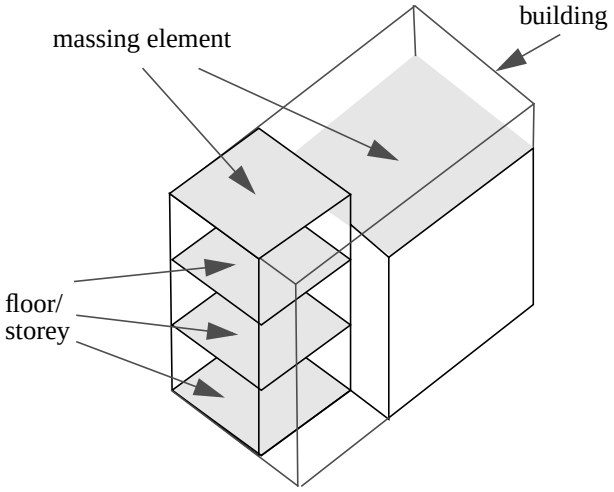


FIGURE 20. Relations between Building FUs, Mass. Element FUs and Floor/Storey FUs

Roof FU A roof is a special floor without constituents. It has two subtypes: Flat Roof FU and Sloped Roof FU. SEED-Layout does not do anything with roofs; it only uses them when it creates a 3-dimensional extrusion of a multi-storey Layout, for example, in order to display it (see Figure 21for an example); it looks at the roof type to find an initial geometric form for a roof.

Horizontal Zone FU A horizontal zone may contain vertical zones like shafts, horizontal zones, or rooms. It may not contain floors, massing elements, or buildings.

Vertical Zone FU A vertical zone may be the constituent of a floor or horizontal zone, provided the horizontal zone it the constituent of a floor. It is expected to be on at least two floors. The floors containing the same vertical zone must form a sequence of floors not interrupted by floors that do not contain the vertical zone. Vertical zones have two

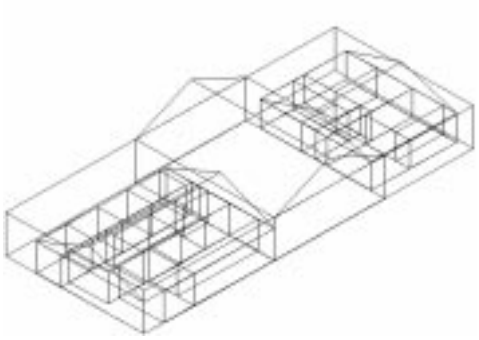


FIGURE 21. Extruded FIRESTATION with sloped roofs

subtypes: Shaft FUs and Atrium FUs (see the Fundamental Concepts Manual for details).

Room FU A room can have no other spatial constituents. That is, it is assumed not to be further subdivided.

Should SEED-Layout be expanded in the future to allow also for the allocation of equipment and furniture in rooms, Room FUs will be allowed to have these types of constituents.

3.8 SEED-Layout Output

At any level, you may save a layout for use in a subsequent session. You may open such a layout file when you have activated exactly the same Layout Problem for which you generated the Layout. Consult the Reference Manual, section 3.1, for details.

Save a Layout:

1. **File > Save Layout ...** opens a file dialog box.
2. Enter the desired file name and click Save.

You may also save a file that is readable by MDS. Consult the Refence Manual, section 3.1, for details.

Save an MDS File:

1. Check if the input and output units are correctly set by opening the Units dialog box: **Edit > Grid/Units.** Edit the units if needed.
2. **File > Save MDS File ...** opens a file dialog box to enter the file name as in the Save Layout command.
3. Enter the desired file name and click Save.

Note that an MDS file saves those and only those Design Units that have not been expanded; that is, they contain no sublayouts. For example, a floor will be saved if it was never expanded; if it was expanded, but only down to the level of zones, the zones will be saved, but not the floor; if some zones were expanded and others were not, some zones will be saved, while others will be replaced by the Design Units inside. It is your responsibility to make sure that all Design Units are expanded to a level where they represent properly named MDS modules.

You may save an .html file that describes the active Layout Problem in a form that can be easily read by HTML browsers and printed. Consult the Refence Manual, section 3.6, for details.

Save Problem Document:

1. Check if the input units are correctly set by opening the Units dialog box: **Edit > Grid/ Units.** Edit the units if needed.
1. **Problem > Save Document File...** opens a file dialog box.
2. Enter or select the desired file name. Do NOT enter the .html extension.
3. Click Save.

3.9 Extended Exercise

Open the San Antonio Training Building problem. Try to lay out as many parts as you feel inclined to and create an MDS file.

4. *Session 3: The SEED-Layout Design Space*

Prerequisites: successful completion of Session 2.

Approximate time to complete:

4.1 Overview

This chapter introduces advanced concepts centered around the notion of “design space” as it is understood in the SEED-Layout context. Once you mastered this material, you may sigh with relief: this is as complicated as it will ever get in SEED-Layout. The following concepts are the focus of the session:

- SEED-Layout problems and subproblems/spaces
- Alternative subproblems/spaces
- Variant problems/spaces
- SEED-Layout design space
- Navigation in the design space

The goal of present session is to make you understand the above concepts and show you how to “navigate” through the SEED-Layout Design space.

4.2 A Simple Design Space

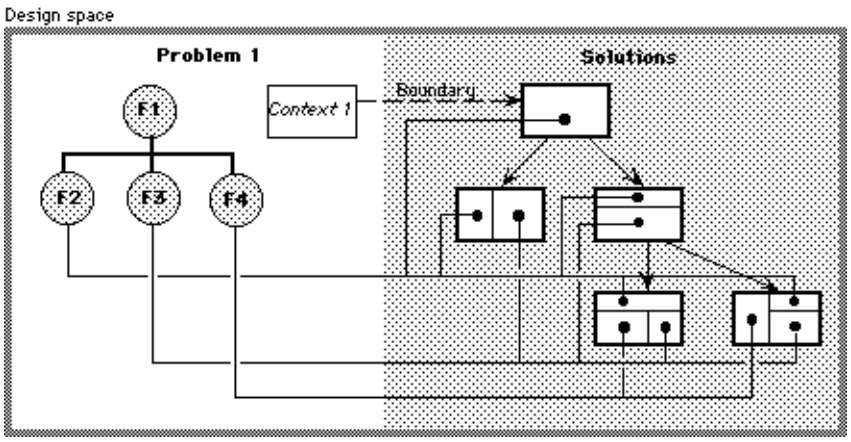


FIGURE 22. A simple design space

Load the FIRESTATION Layout Problem known to you from Session 2. We are going to retrace some of the exercises from Session 2, but pay more attention to the multiple relations between elements established in this process. In fact, we will introduce only a few new commands in the present session; the focus is on arriving at a deeper understanding of the underlying concepts.

Start by allocating again the three top-level massing elements. Inspect the parent/child relations between the Layouts in the Layout Tree Window. In addition, clarify in your own mind the relations between the Design Units in the Layouts and the Functional Unit constituents of the Layout Problem that are associated with the Design Units. These relations are depicted abstractly in Figure 22. To make this and the following figures easier to understand, we drew Functional Units that occur in more than one space individually for each space; a grey link between two elements indicates that the two units are in fact the same. Solid and dashed black links, in contrast, indicate relations between distinct elements. Note the one-to-many relations between parent and child layouts, on the one hand, and between Functional Units and Design Units, on the other hand. All of these relations are explicitly managed by SEED-Layout, and you do not have to remember them.

We call the Layout Problem and the Layout solutions generated for this problem a *design space*. Every design space contains at least one Layout and ‘knows’ the last active Layout.

4.3 Subproblems and Subspaces

Expand the administrative wing. This allocates the first storey in this wing. (Optional) allocate the adm_wing_roof on top of the first storey.

Expand the first floor in the administrative wing. This allocates the day zone in this wing. Place the administrative zone beneath the day zone. Open the Layout Problem Manager:

Window > Layout Problem Manager.

This is a window we have not seen in prior sessions. It depicts explicitly the relations between super- and subproblems and allows you to navigate between problems making larger steps than are possible with the Goto commands and navigation buttons introduced in Session 2:

Activate a Layout Problem in the Layout Problem Manager:

1. Select a problem in the Layout Problem Manager by clicking the left mouse button on it.
2. Click the right mouse button to open a pop-up menu and select the Activate option. This changes the currently active Layout Problem and reactivates the last active Layout in the associated design space.

Try to understand the relations between the problems and subproblems and the design spaces to which they belong. Figure 23illustrates the space/subspace relation at one level. Observe that in our current example, this relation occurs twice:

- between the design space FIRESTATION and its subspace administrative wing
- and between the design (sub)space administrative wing and its subspace first floor.

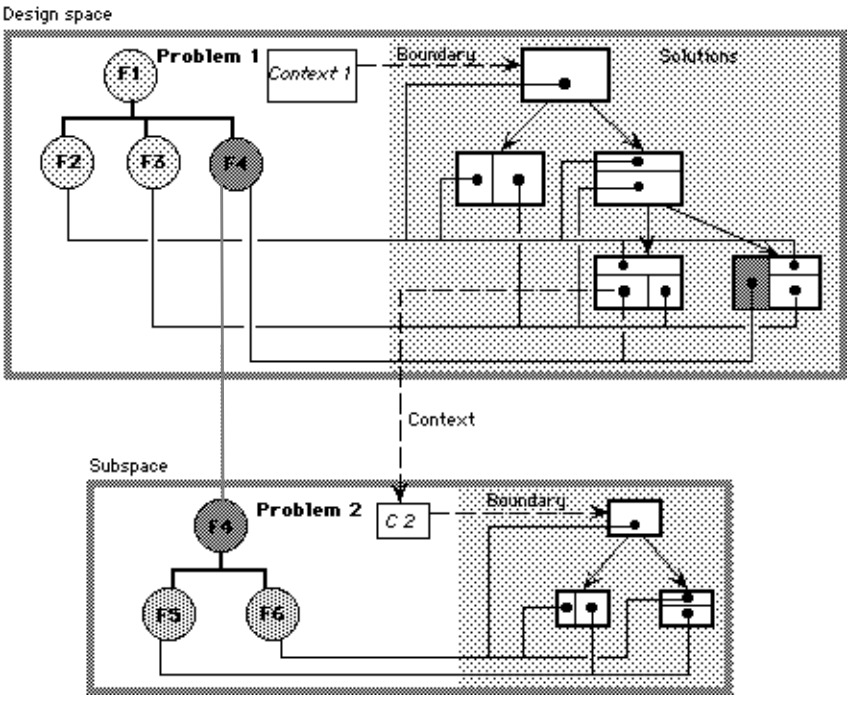


FIGURE 23. Subproblem and subspace relations

Note that a subspace has the same kinds of parts as the superspace to which it belongs. Observe how the Context for a subproblem is determined by a Design Unit in the superspace.

In general, space/subspace relations can be arbitrarily deep in SEED-Layout. The depth that can be reached for a given Layout Problem is determined by the depth of the constituent hierarchy. But remember that a Functional Unit and its constituents do not define a Layout Problem by themselves: a Context is needed in addition. It is in fact conceivable that the same Functional Units occur in different buildings or projects. It is always the Context that makes the task of allocating Functional Units specific.

4.4 Alternative Subproblems and Subspaces

Revisit the first floor in the administrative wing. Expand the day zone and allocate the constituents in it.

Goto the superproblem first floor. Create an alternative layout of the two zones by clicking the Next Sibling button. Expand the day zone again and allocate the constituents in it.

Inspect the Layout Problem Manager. Why does it show two Layout Problems called dayZ_1? Visit and revisit the two expanded layouts of dayZ_1 (make sure the superlayout view is on). Clearly, each allocates the same constituents, but the Contexts are different (see also Figure 24).

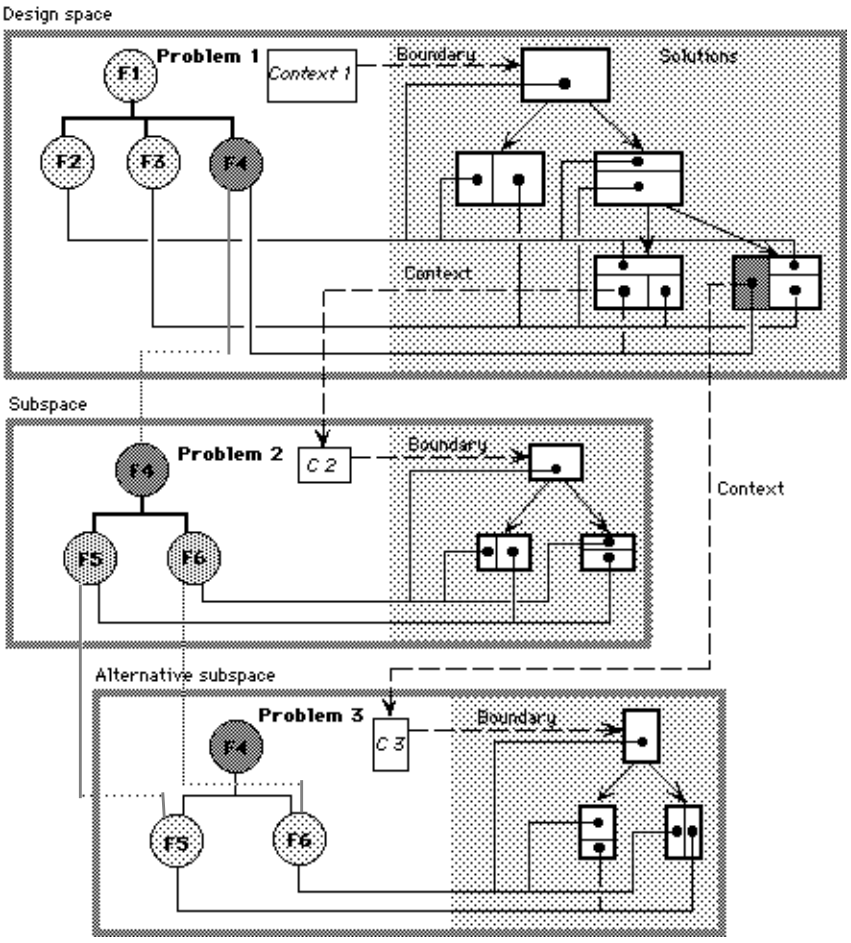


FIGURE 24. Alternative subproblems and subspaces

Recall from Session 1 that a Layout Problem has two basic parts: a Context, a top functional Unit with its constituents and associated requirements. If we change any one of these parts, we create a new Layout Problem. The Layout Problems that we created by expanding the day zone in two different layouts share the top Functional Unit and constituents, but each has a unique Context. The two Layout Problems and the associated design spaces are therefore different.

We call them *alternative subproblems* and *subspaces* because they are at the same level in the problem hierarchy, but indicate alternative solutions to the overall Layout Problem.

4.5 Variant Layout Problems and Design Spaces

Revisit the first floor in the administrative wing. Open the Constituent Hierarchy Window and inspect its contents. Reopen the Layout Problem Manager (if it has been closed).

Select the day zone in the active Layout and click the Disaggregate button in the command bar of the Design Window. Observe what happens in the Design Window, the Constituent Hierarchy Window, and the Layout Problem Manager.

We changed the constituents of the Layout Problem while keeping the Context constant. This is the opposite effect of creating an alternative subproblem. We nevertheless created a different Layout Problem because the constituents of the top Functional Unit, the first floor in the administrative wing, have changed. To distinguish this case from the case of alternative subproblems, we call the new Layout Problem a *variant* of the old one. Figure 25 depicts the relations between all elements discussed in this session.

SEED-Layout Design Space We call all problems, subproblems, alternative problems, variant problems and the layouts generated in each of the associated design spaces the (overall) ***SEED-Layout design space***.

Finalize through all levels. Navigate through the problem hierarchy and observe how the sublayouts again fit into their superlayouts.

Picture in your mind what SEED-Layout had to do when it finalized the overall Layout: It not only had to link all super- and sublayouts together, it also had to make sure that at each sublevel, it used the last active design space variant.

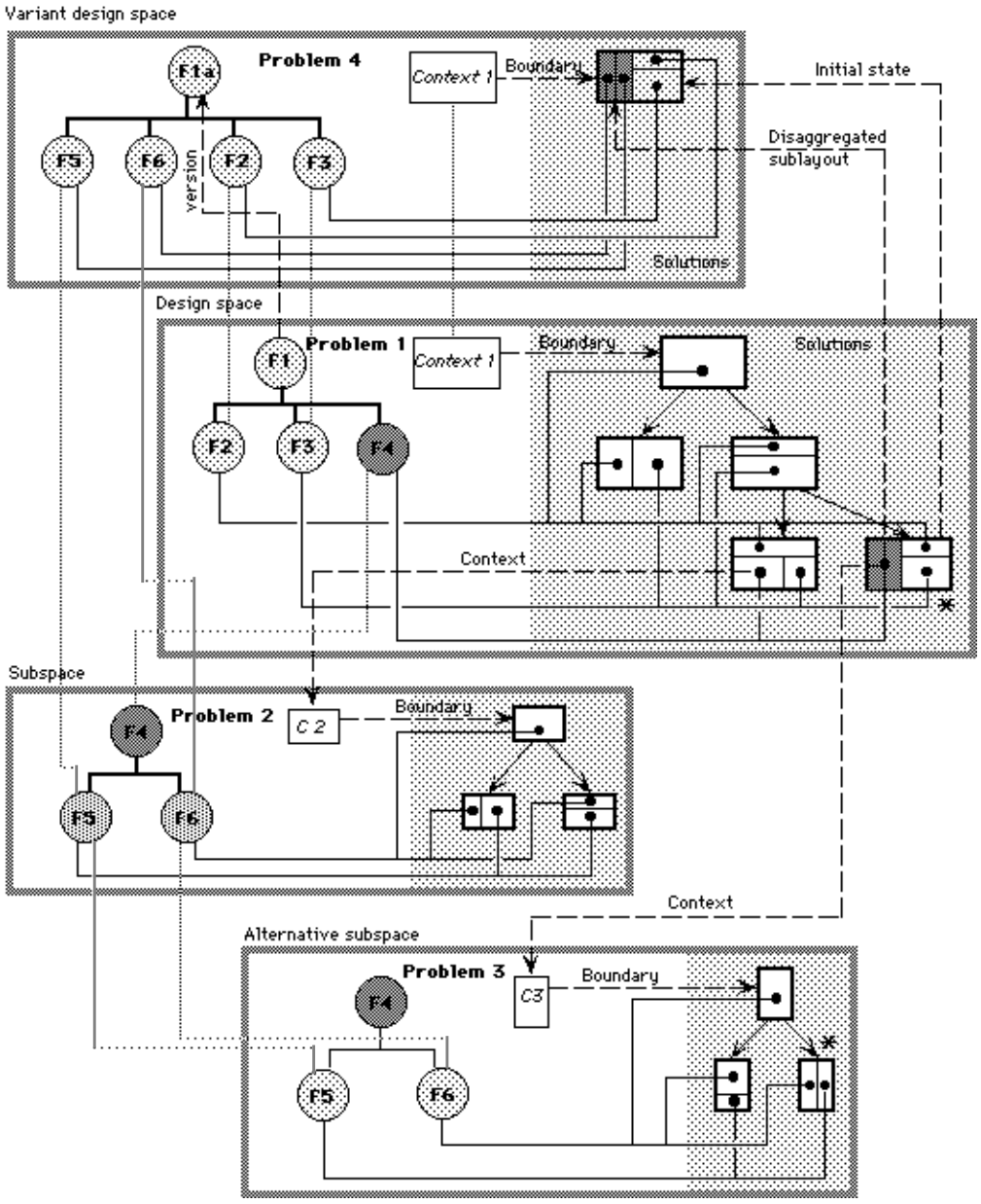


FIGURE 25. The full SEED-Layout design space

Exercise:
Revisit each subspace whose active Layout was finalized and reactivate its parent. Revisit the first floor variant design space, which contains the disaggregated Design Units from the former day zone. Select the four walls that surround these Design Units and click the Aggregate button in the command bar.

This opens a dialog box which prompts you for a Functional Unit name. Key-in a name and click OK.

Observe again in the open windows what happens and try to understand the changes that you see in light of what you have learned about the disaggregate command.

4.6 Extended Exercise

Open a Layout Problem describing the San Antonio training building and try to generate a complete layout of all massing elements and floors inside each massing element.