massachusetts institute of technology — artificial intelligence laboratory

The Architect's Collaborator: Toward Intelligent Tools for Conceptual Design

Kimberle Koile

AI Technical Report 2001-001 January 2001

© 2001 massachusetts institute of technology, cambridge, ma 02139 usa — www.ai.mit.edu

The Architect's Collaborator:
Toward Intelligent Tools for Conceptual Design

by

Kimberle Koile

Submitted to the Department of Electrical Engineering
and Computer Science in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2001

© Massachusetts Institute of Technology, 2001. All Rights Reserved.

The Architect's Collaborator:
Toward Intelligent Tools for Conceptual Design

by

Kimberle Koile

Submitted to the Department of Electrical Engineering and
Computer Science on February 5, 2001, in partial fulfillment of the
requirements for the degree of Doctor of Philosophy

Abstract

In early stages of architectural design, as in other design domains, the language used is often very abstract. In architectural design, for example, architects and their clients use experiential terms such as "private" or "open" to describe spaces. If we are to build programs that can help designers during this early-stage design, we must give those programs the capability to deal with concepts on the level of such abstractions. The work reported in this thesis sought to do that, focusing on two key questions: How are abstract terms such as "private" and "open" translated into physical form? How might one build a tool to assist designers with this process?

The Architect's Collaborator (TAC) was built to explore these issues. It is a design assistant that supports iterative design refinement, and that represents and reasons about how experiential qualities are manifested in physical form. Given a starting design and a set of design goals, TAC explores the space of possible designs in search of solutions that satisfy the goals. It employs a strategy we've called dependency-directed redesign: it evaluates a design with respect to a set of goals, then uses an explanation of the evaluation to guide proposal and refinement of repair suggestions; it then carries out the repair suggestions to create new designs.

A series of experiments was run to study TAC's behavior. Issues of control structure, goal set size, goal order, and modification operator capabilities were explored. In addition, TAC's use as a design assistant was studied in an experiment using a house in the process of being redesigned. TAC's use as an analysis tool was studied in an experiment using Frank Lloyd Wright's Prairie houses.

Thesis Supervisor: Randall Davis
Title: Professor

Acknowledgments

It's about adventure.
— Richard Feynman

Researching and writing this thesis has certainly been an adventure—in AI, in architecture, in remembering to eat and sleep. I am profoundly grateful to those who helped me along the way.

My Committee:
Randy Davis, Aaron Fleisher, Howie Shrobe, Patrick Winston

AI:
Randy Davis, Howie Shrobe, Patrick Winston, John Aspinall, Tomás Lozano-Peréz

Architecture:
Aaron Fleisher, Richard Krauss, Duncan Kincaid, Mark Gross, Stephen Ervin, Bill Mitchell, Bill Porter, Don Schön, The Architects Collaborative

Life:
John and Michael Aspinall, Earl and Carmon Koile, Kristen Wells, Stephen Koile, Scott McKay and Laura Need, Andee Rubin and Liz Bradley, Annie LaCourt and Mark Burstein, Una-May O'Reilly, Lisa Tucker-Kellogg, Sue Felshin, Larry Hunter, Noel Chiappa, Susan Wilson

Table of Contents

1 Introduction ... 13
1.1 The Problem ... 13
1.2 Our Solution ... 15
1.3 An Example ... 16
1.4 Motivation ... 25
1.5 Guide to This Document ... 29
2 Approach ... 31
2.1 The Design Process ... 31
2.2 A Design Assistant ... 32
2.3 TAC as Design Assistant ... 32
    2.3.1 Dependency-Directed Redesign ... 33
    2.3.2 Intelligence ... 35
3 Defining a Design Problem ... 39
3.1 Representing Designs ... 39
3.2 Representing Design Goals ... 44
    3.2.1 Design Characteristics ... 44
    3.2.2 TAC-Functions ... 50
3.3 Design Problem Example ... 51
4 Evaluating Design Goals ... 53
4.1 Examples ... 53
4.2 Explanations ... 57
5 Suggesting Repairs ... 61
5.1 Types of Suggestions ... 61
5.2 Representing and Using Repair Knowledge ... 63
    5.2.1 Fixers and Setters ... 64
    5.2.2 Increasers, Decreasers, and Influences ... 67
5.3 Compound Suggestions ... 71
5.4 More Examples of Suggesting Repairs ... 71
6 Performing Repairs ... 85
7 Control Structure ... 91
7.1 Overview ... 91
7.2 Goal Interaction: Conflict and Synergy ... 92
7.3 The Control Structures ... 96
    7.3.1 Sequential-with-Lookahead ... 96
    7.3.2 Sequential ... 119
    7.3.3 Concurrent ... 128
7.4 Goal Order ... 143
7.5 Termination ... 147
    7.5.1 Identifying Equivalent Designs ... 147
    7.5.2 Limiting Iteration ... 149
7.6 Summary ... 151
7.7 Experiments ... 153

8 Exercises in Architecture 169
8.1 Design: Chatham House 169
8.2 Analysis: Prairie Houses 204
9 Related Work 211
9.1 Reasoning Methodologies 211
9.1.1 Planning 211
9.1.2 Case Adaptation 214
9.1.3 Performance-based Refinement 216
9.2 Experiential Knowledge 217
10 Conclusion 221
10.1 Future Work 221
10.2 Summary 223
References 225
R.1 Bibliography 225
R.2 Illustration Credits 230
Appendices 231
A Diagram Conventions 231
B Knowledge Base 233
C Language Terms 241
D Alternate Territory Models 243
E Explanation Examples 245
F Modifying Edge and Territory Models 251
G Details of Architecture Exercises 253

List of Figures

1.1 Classroom ... 13
1.2 Studio at Taliesin West, by Frank Lloyd Wright ... 14
1.3 Living room ... 14
1.4 Living room in the Hanna house, by Frank Lloyd Wright ... 15
1.5 Design example: first floor, second floor, section through stair. ... 17
1.6 TAC computes and displays the regions created by the physical form. ... 18
1.7 The shaded area is visible from the Living region; * represents a viewpoint ... 18
1.8 Shaded area is visible from the front door; * represents a viewpoint at front door. ... 19
1.9 Verifying that a screen (in lower design) decreases visual openness ... 20
1.10 Two new designs: each has stair rotated to increase visual openness ... 21
1.11 Visible areas and visual openness measures for design #2. ... 22
1.12 Visible areas and visual openness measures for front door in design #2. ... 22
1.13 Comparison of stair access path lengths and changes in direction ... 23
1.14 Rejected new designs with the stair on an exterior edge. ... 24
3.1 Partial design element model (left) and derived edge model (right). ... 40
3.2 TAC's input is in the form of a design element model and an edge model. ... 41
3.3 Mrs. Thomas Gale house, Oak Park, Illinois (1904, 1909) ... 42
3.4 Models for the Mrs. Thomas Gale house. ... 43
3.5 Shaded region of Gale Living territory is visible from Dining territory; ... 45
3.6 Living territory of Gale house and view to Dining territory. ... 46
3.7 Path for calculating change in direction from Gale Front door to Living territory. ... 47
3.8 Dependency links for example design characteristics. ... 50
4.1 Floorplan and territory model for Tomek house main (second) floor. ... 53
4.2 Shaded region of Living territory is visible from Dining territory. ... 54
4.3 Living territory of Tomek house; view to Dining territory. ... 55
4.4 Floorplan showing Gale doors used for `perceived-main-entryness` comparison. ... 55
4.5 Path from usual approach point (o) to Gale Front door. ... 56
4.6 Explanation for `(visually-open Living from Dining)`. ... 57
4.7 Explanation for `physical-accessibility` expression. ... 59
5.1 Territory model for Tomek house main (second) floor. ... 62
5.2 "Punctured" fireplace in the Robie house, designed by Frank Lloyd Wright. ... 63
5.3 Searching for fixers in the explanation ... 65
5.4 Explanation for `(visually-open Living from Dining)`. ... 68
5.5 Floorplan and territory model for Horner house main (first) floor ... 72
5.6 Part of explanation for `(on-interior-edge fireplace Living)`. ... 74
5.7 How `some-fixer` proposes moving the fireplace or adding a new fireplace. ... 75
5.8 Two new Horner designs with a fireplace on an interior edge. ... 76
5.9 The Chatham house and approach paths to exterior doors. ... 77
5.10 Directions of influence for `perceived-main-entryness` ... 78
5.11 New Chatham designs with Front door having more perceived-main-entryness. ... 81
6.1 Tomek territory models: with fireplace and without. ... 86
6.2 Territory model for Chatham house, first floor. ... 87
6.3 New Chatham design with stair rotated. ... 87

6.4 Bounds for Living and Dining territories. 88
6.5 Rotating a stair without removing projected edges. 89
7.1 Sequential control structure; lookahead is used to propose suggestions in step 1. 97
7.2 Suggestion proposal using lookahead.......... 98
7.3 Lookahead results for `visually-open` and `fireplace-count` goals. 101
7.4 Tomek#1 and region of Living territory visible from Dining territory. 102
7.5 Lookahead results for `visually-open` and `visible-center` goals. 104
7.6 New designs for Tomek `visually-open` and `visible-center` goals. 104
7.7 Floorplan and territory model for Horner house main (first) floor.......... 105
7.8 Five designs with fireplace moved to interior edge, one with new fireplace added.......... 107
7.9 Horner#1: one of the designs to be repaired.......... 108
7.10 Lookahead results for `visually-open` and `fireplace-on-interior-edge` goals. 110
7.11 Repair of Horner#1 results in four new designs which are solutions. 111
7.12 Horner#1#3, with fireplace moved to interior edge and punctured.......... 112
7.13 Horner#6: a new fireplace has been added to Horner. 112
7.14 Territory model for Horner#1.......... 114
7.15 Lookahead results for `fireplace-on-exterior-edge` goal. 116
7.16 TAC checks for conflict between suggestions for current goal and other goals. 117
7.17 New designs with fireplace on exterior edge.......... 117
7.18 Sequential control structure. 120
7.19 Suggestion proposal for the sequential control structure.......... 121
7.20 Designs to be repaired so that `visually-open` goal is satisfied. 124
7.21 Design to be repaired to have one fireplace. 126
7.22 Concurrent control structure.......... 129
7.23 Concurrent-lookahead 130
7.24 Pruning of suggestions for `visually-open` and `visible-center` goals.......... 135
7.25 Territory model for Horner#1.......... 137
7.26 Territory model for Horner#1.......... 143
7.27 Same designs generated via different modification steps.......... 147
7.28 Comparison of solutions with stair punctured in slightly different locations.......... 148
7.29 Two designs stopped at iteration limit with sequential control structure. 150
7.30 Territory model for Horner.......... 153
7.31 Four solutions for the control structures with optimal goal order. 157
7.32 Five extra solutions found for sequential-with-lookahead and concurrent control structures with nonoptimal goal order.......... 159
7.33 Comparison of two solutions for sequential control structure.......... 160
7.34 Extra new solutions for sequential control structure with nonoptimal goal order.......... 161
7.35 Two extra solutions for control structures with move operator. 163
7.36 Examples of extra solutions for concurrent with move operator.......... 164
7.37 Examples of extra solutions for sequential with move operator. 165
7.38 Nonminimal solution avoided by sequential with lookbehind. 167
8.1 The Chatham house. 169
8.2 The Chatham house first floor and approach paths to exterior doors.......... 170
8.3 View from front door into living room.......... 171
8.4 View from living room to dining room. 171
8.5 Territory model and approach paths for Chatham.......... 174

8.6 Designs TAC proposes with front door as perceived main entry. 175
8.7 Designs TAC proposes with side door as perceived main entry. 176
8.8 Two designs: perceived main entry is front door (left) or side door (right). 177
8.9 Chatham#2 visual openness and accessibility of Living from front door. 177
8.10 Decreasing visual openness by replacing open edges with walls. 179
8.11 Decreasing visual openness by replacing open edges with screens. 180
8.12 Designs with visual openness repaired to be less than 0.8. 181
8.13 Design derived from #1, path change in direction has been repaired. 182
8.14 New designs for Chatham#2#2 after repairing for visual openness > 0.3. 184
8.15 New designs for Chatham#2#3 after repairing for visual openness > 0.3. 185
8.16 Starting design (#2), solutions for Living visual openness and path from front door. 186
8.17 Starting design with side door as perceived main entry. 187
8.18 Chatham#10 visual openness and accessibility of Living from side door. 188
8.19 New designs with visual openness increased between Living and side door. 189
8.20 Another design with visual openness increased between Living and side door. 190
8.21 New designs left after pruning those whose intended goal was not met. 190
8.22 Decreasing change in direction between Living territory and side door. 191
8.23 Starting design (top), solutions for Living visual openness and path from side door. 192
8.24 Visual openness of Dining territory from Living territory; value is 0.42. 193
8.25 Designs that increase visual openness of Dining from Living. 194
8.26 More designs that increase visual openness of Dining from Living. 195
8.27 Screenifying the stair: a wall has been replaced with a screen in the Chatham house. 196
8.28 Use space model for Chatham#10#1#4: territories carry indication of intended use. 196
8.29 New designs with kitchen activity adjacent to dining activity. 197
8.30 Architects' design (top) and two of TAC's designs. 200
8.31 Architects' alternate design (top) and two of TAC's designs. 201
8.32 Prairie and Transition House data sets; * indicates main living space. 206
8.33 Non-Prairie House data set; * indicates main living space. 207
8.34 Experimental results for Prairie, Non-Prairie, and Transition houses. 208
A.1 Floorplan drawing. 231
A.2 Model for floorplan shown in Figure A.1. 232
D.1 Territory model for Tomek house first floor. 243
D.2 Alternate territory model for Tomek house first floor. 243
E.1 Three explanation templates. 246
F.1 Portion of territory model for Chatham. 251
F.2 Portion of edge model for Chatham. 251
F.3 Remove stair's projected edges. 251
F.4 Carry out modification: rotate stair. 252
F.5 Add projected edges for stair and walls. 252
F.6 Identify new territories by finding closed polygons in edge model 252
G.1 Designs TAC proposes with front door as perceived main entry. 254
G.2 Designs TAC proposes with front door or side door as perceived main entry. 255
G.3 Designs with side door as perceived main entry, cont'd. 256

List of Tables

7.1 Comparing control structure features. ... 151
7.2 Summary of experiment parameters. ... 155
7.3 Results of experiments with optimal goal order. ... 156
7.4 Results of experiments with nonoptimal goal order. ... 158
7.5 Results of adding move to exterior edge as means of increasing visual openness; optimal goal order. ... 162
7.6 Results of adding move to exterior edge as means of increasing visual openness; nonoptimal goal order. ... 163
7.7 Results of experiments with lookbehind and nonoptimal goal order. ... 166

Chapter 1

Introduction

The Architect's Collaborator (TAC) is a prototype design support system for early stages of architectural design. It supports iterative design refinement, representing and reasoning about how experiential qualities are manifested in physical form. It assists a designer in specifying goals and in exploring the space of possible designs. It evaluates design goals, proposes and refines repair suggestions for unsatisfied goals, and carries out those suggestions to create new designs.

The Architect's Collaborator was built in order to answer two questions:

- How are experiential qualities translated into physical form?
- How might a tool assist designers with this process?

Before discussing TAC's answers, it's important to know what we mean by experiential qualities and by their being manifested in physical form.

1.1 The Problem

By *experiential qualities* we mean qualities such as openness or privacy, concepts that architects and their clients use when describing buildings and spaces. By *physical form* we mean the design elements—walls, windows, doors—that create buildings and spaces. Experiential qualities and physical form are intimately related: The form that creates a space shapes the way in which we experience that space. This shaping of experience is what we mean by experiential qualities being manifested in physical form. Here are some examples.

Imagine sitting at one of the desks shown in Figure 1.1.

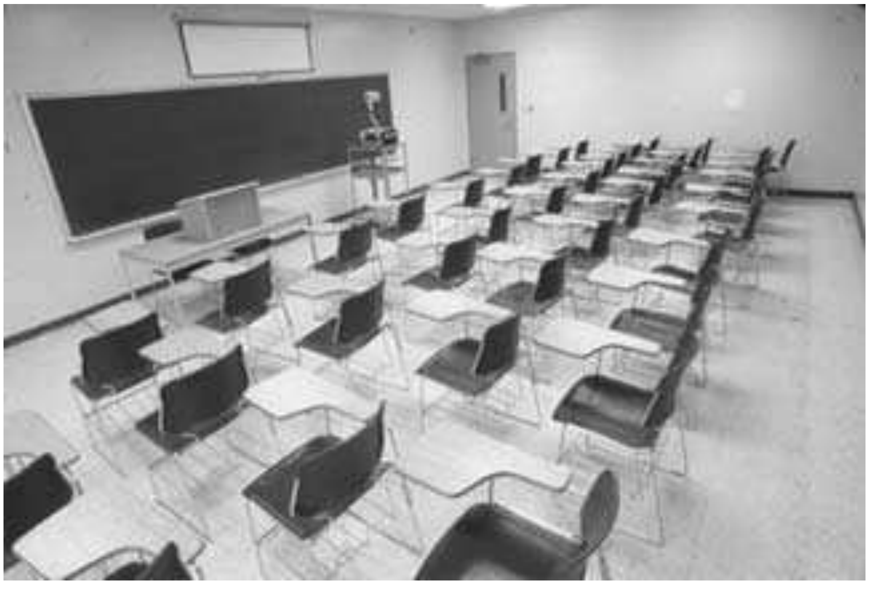


Figure 1.1: Classroom.

Now imagine sitting at one of the desks shown in Figure 1.2, and ask yourself why your experience would be so different.

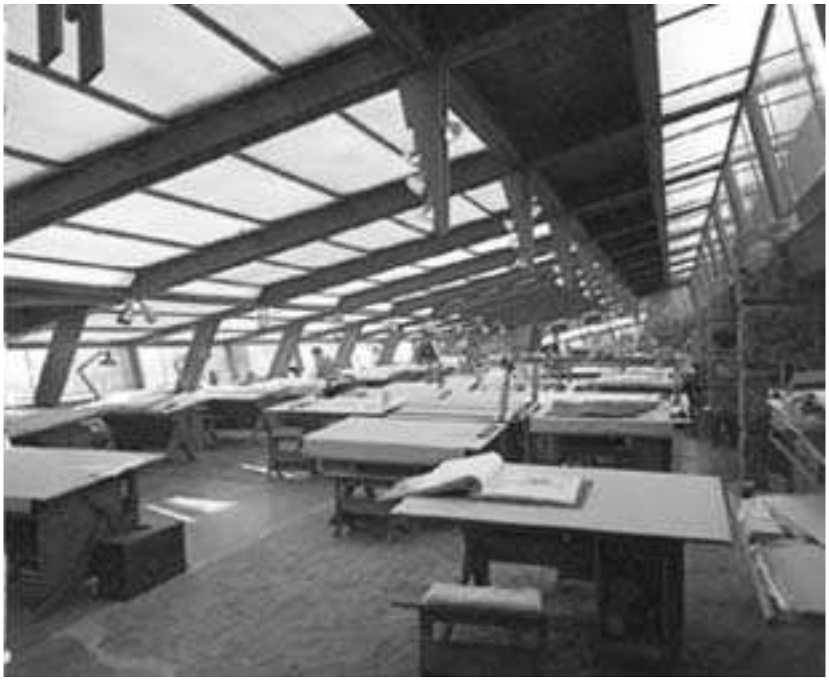


Figure 1.2: Studio at Taliesin West, by Frank Lloyd Wright

You'll invariably think about the height of the ceiling, the light from above, the natural wood. All these aspects shape your experience, and not accidentally. Frank Lloyd Wright intended and expected for you to be profoundly influenced by this environment.

Looking at another example, imagine how it would feel to sit in the living room shown below.

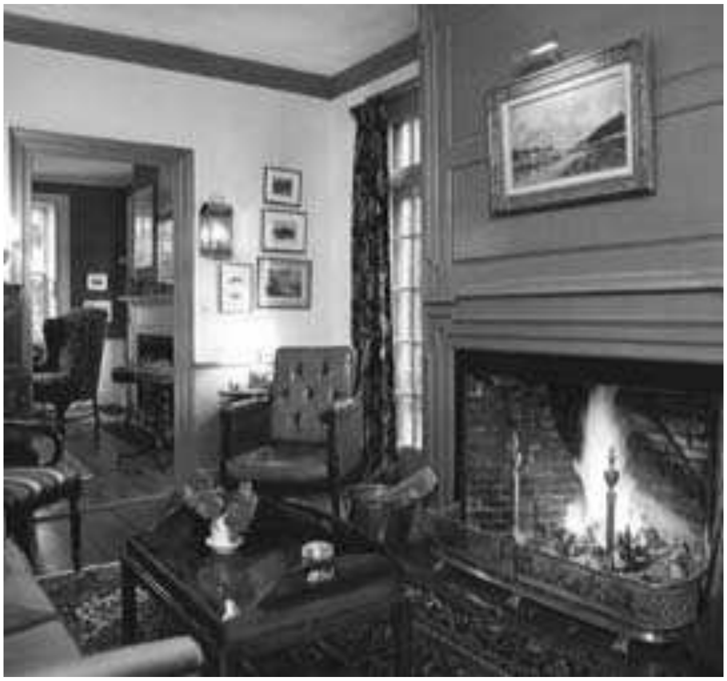


Figure 1.3: Living room.

Now imagine sitting in the living room shown in Figure 1.4. Why does this living room feel different?

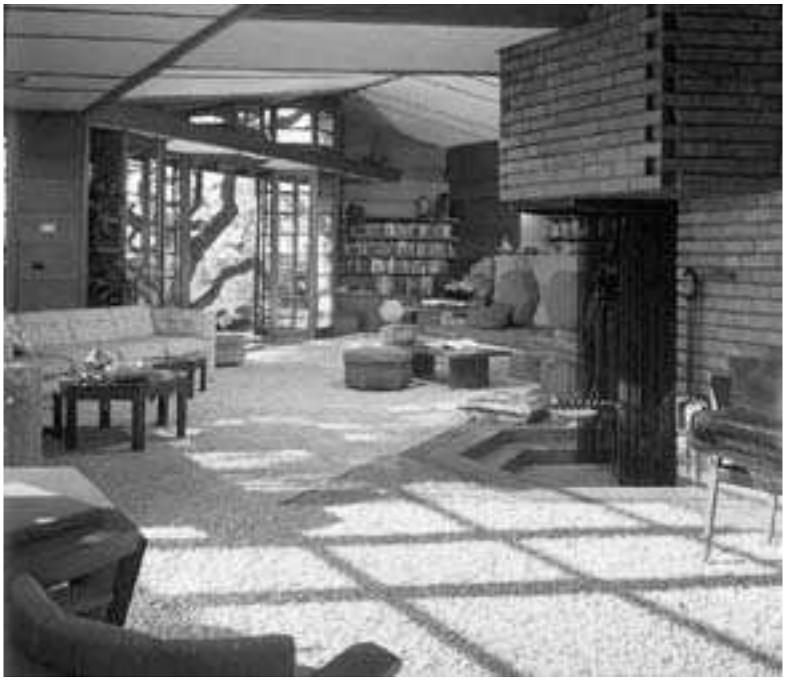


Figure 1.4: Living room in the Hanna house, by Frank Lloyd Wright

Two living rooms; two very different uses of materials, light, arrangements of walls and windows; and two very different experiences.

As these two examples have illustrated, experiential qualities of a space are manifested in the physical form that creates that space. What's more, those experiential qualities are paramount in architectural design. As Frank Lloyd Wright said of another of his buildings:[1]

> ...the reality of the building did not consist in the walls and in the roof, but in the space within to be lived in.

1.2 Our Solution

In the course of building The Architect's Collaborator we discovered how experiential qualities could be translated into physical form, and how a tool could assist designers with this process.

- How are experiential qualities translated into physical form?

TAC's languages and representations formalize abstract concepts such as openness and privacy, and relate those concepts to arrangements of physical form.

- How might a tool assist designers with this process?

TAC enables designers to specify design goals and explore the space of possible designs satisfying those goals. TAC employs what we've called *dependency-directed redesign*: it evaluates design goals, proposes and refines repair suggestions, and creates new designs by carrying out those suggestions. By means of dependency-directed redesign, TAC is able to deal with issues of a very large search space, noninvertible evaluation functions, complex interactions between design modification operators, and multiple conflicting goals.

---

[1] Unity Temple in Oak Park, Illinois.

1.3 An Example

We envision The Architect's Collaborator being used in the following fashion.

> Imagine that you're sketching a design, pen in hand. You tell the computer near you that the design is for a client who wants a house with main living spaces that feel open to one another, but private with respect to the front door. You have ideas about physical forms that manifest feelings of privacy and others that manifest feelings of openness, and you're translating those ideas into lines and annotations on a page. You stop to assess your latest sketch and ask the computer for its comments. It shows you regions defined by your proposed physical forms. It shows you what is visible from each region and from the front door. You notice that a large portion of the living room is visible from the front door, so you add a screen; the computer shows you what is now visible with a screen in place. You also notice that the stair blocks the view of the dining room from the living room. You ask the computer to increase the visual openness of the dining room without sacrificing privacy with respect to the front door or ease of access to the stairs. It suggests two other locations for the stair, showing you how the visual openness, privacy, and access are affected by each location. It shows you several other possible stair locations, but points out that in these cases, while visual openness increases, privacy of the dining room decreases, and the stair is not as easily accessed. You like one of the first two proposed locations, accept it, and continue sketching and consulting with your computer.

As shown in the following example, The Architect's Collaborator is able to perform all the steps in the above scenario, apart from operating on a sketched design. It:

1. displays regions defined by the physical form
2. computes and displays visual openness measurements for all regions
3. computes and displays visual openness measurements for the living region from the front door, with and without a screen[2]
4. proposes repair suggestions for increasing visual openness between dining and living
5. creates new designs by carrying out repair suggestions
6. evaluates all goals to check for solutions
7. displays rejected designs

---

[2] By "screen" we mean a perforated structure such as a stair railing or a bookcase without a back. An example of a screen is shown in Figure 8.27.

Consider the design below.

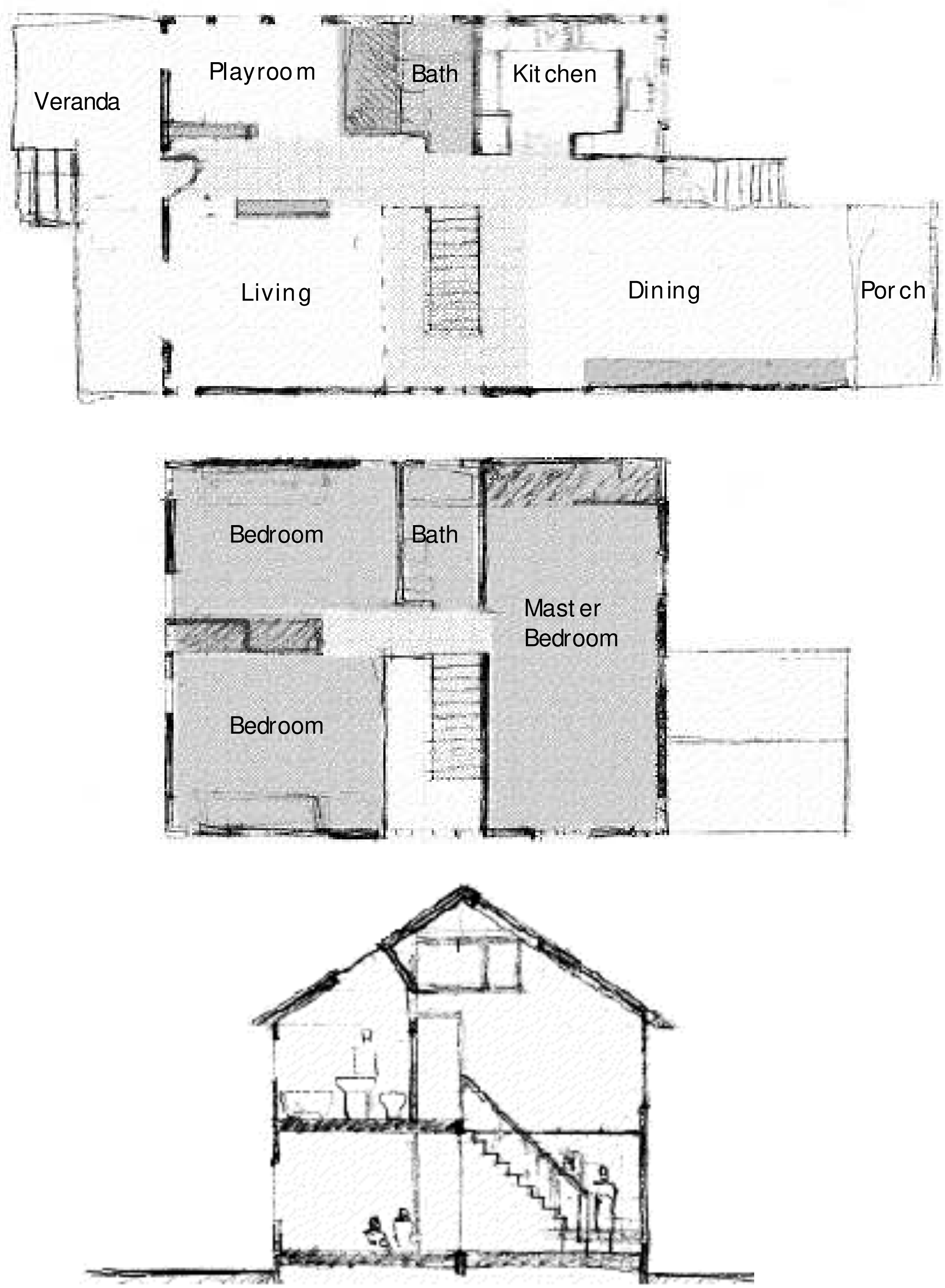

Veranda
Playroom
Bath
Kitchen
Living
Dining
Porch
Bedroom
Bath
Master Bedroom
Bedroom

Figure 1.5: Design example: first floor, second floor, section through stair.[3]

---

[3] Drawings courtesy of Duncan Kincaid and Daniel Gorini. See Appendix A for a description of diagram conventions used throughout this document.

1. <u>Displaying regions</u>

We consider the first floor of this design. The designer asks TAC to display the regions defined by the physical form,[4] then he supplies region names based on intended use.

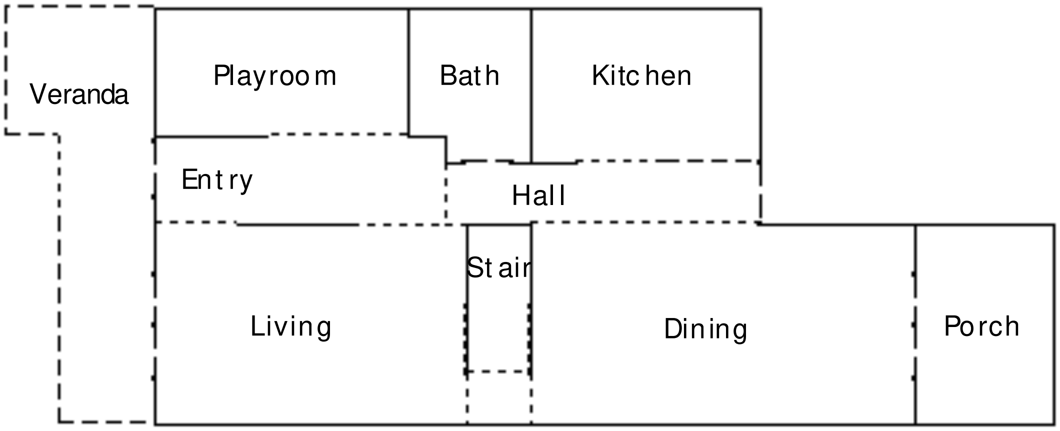
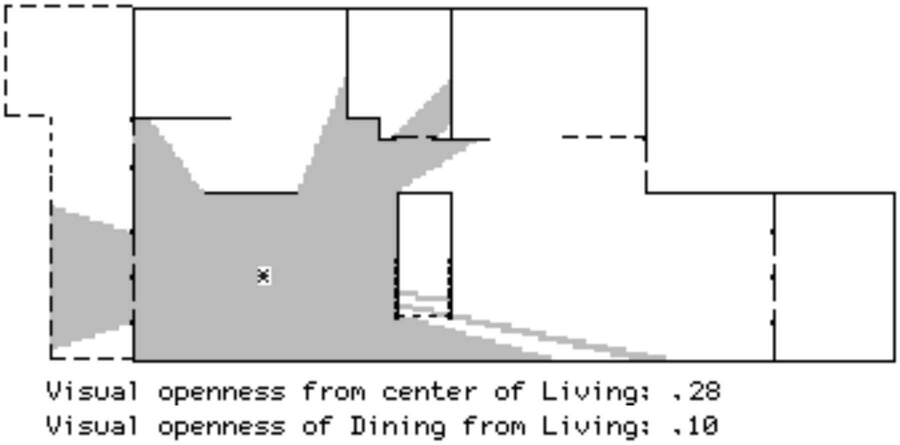


Figure 1.6: TAC computes and displays the regions created by the physical form.

2. <u>Computing and displaying visual openness measurements</u>

The designer is interested in visibility issues and asks TAC to display what is visible from each region and from the front door. TAC responds by computing and displaying visual openness measures, which represent the portion of an area that is visible from a particular location. The portion of the design visible from the Living region is shown below.



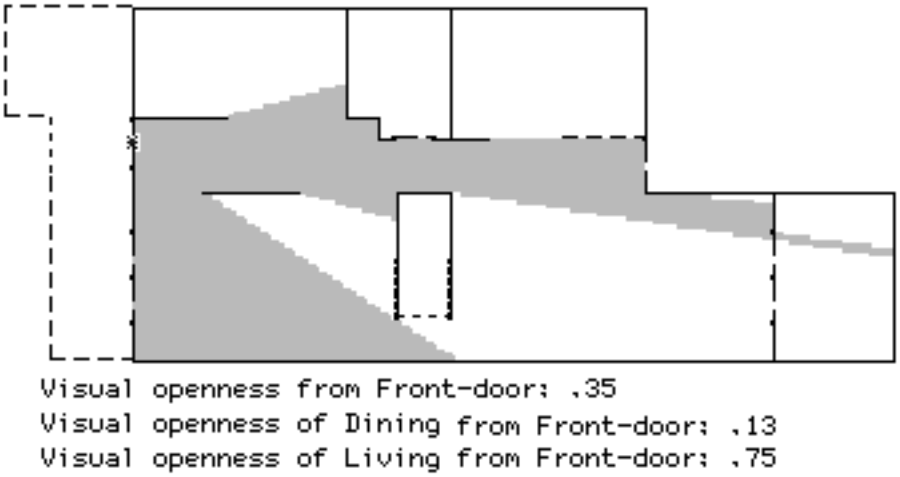
```
Visual openness from center of Living: .28
Visual openness of Dining from Living: .10
```

Figure 1.7: The shaded area is visible from the Living region; * represents a viewpoint at the center of the region. The first visual openness measure represents the portion of all regions visible from the viewpoint; the second measure represents the portion of the Dining region visible from the viewpoint.

---

[4] A note about TAC's use of 2D floorplans: TAC is interesting because of its design capabilities, not its visualization capabilities. More sophisticated visualization capabilities (e.g. 3D, virtual reality) are possible, but are outside the scope of this research.

The figure below shows the area visible from the front door and gives the visual openness measurements that represent the portion of Living and Dining regions visible from the front door.

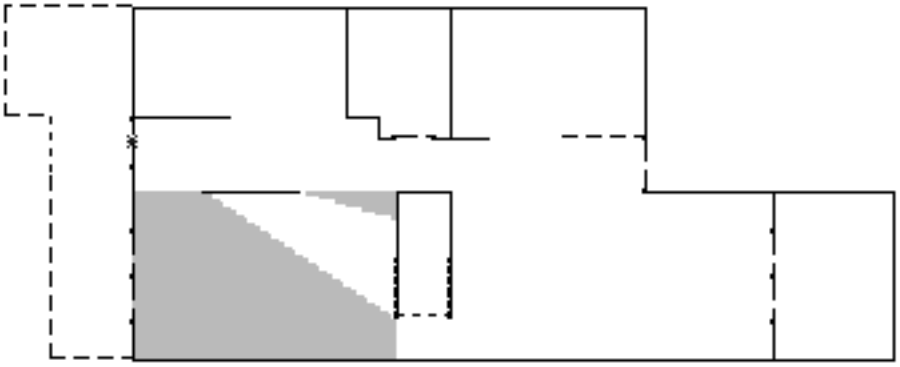


```
Visual openness from Front-door: .35
Visual openness of Dining from Front-door: .13
Visual openness of Living from Front-door: .75
```

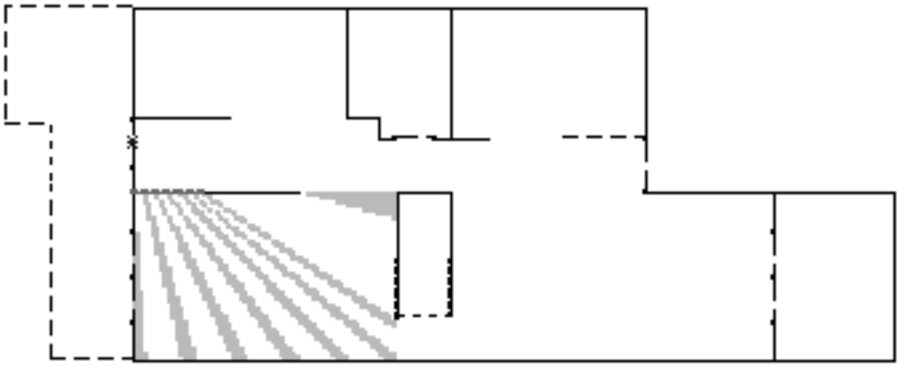
Figure 1.8: Shaded area is visible from the front door; * represents a viewpoint at front door.

3. <u>Computing and displaying visual openness measurements for Living from front door with and without an added screen</u>

The designer notices that a large portion of the Living region is visible from the front door and asks TAC to add a screen between the Entry and Living regions and then to calculate the visual openness of the Living region from the front door for the new design. TAC verifies that the visual openness decreases. (Alternatively, the designer could ask TAC to decrease the visual openness between the Living and Entry territories, and TAC would suggest adding the screen.) The original design and the new design, along with their visual openness measurements, are shown in Figure 1.9.



Visual openness of Living from Front-door: .75



Visual openness of Living from Front-door: .27

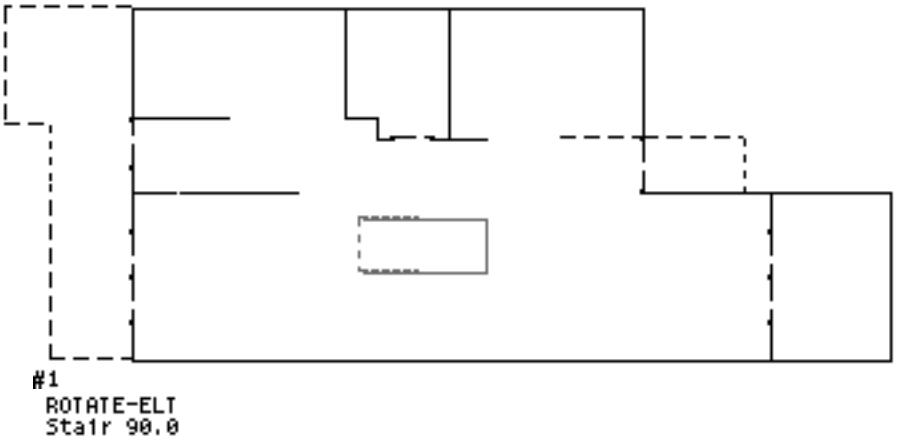
Figure 1.9: Verifying that a screen (in lower design) decreases visual openness of the Living region from front door; * represents viewpoint at front door.

4. Proposing repair suggestions for increasing visual openness of Dining from Living

The designer notices that a stair blocks much of the view of the Dining region from the Living region. He asks TAC to suggest ways to make the Dining region visually open from the Living region, while keeping both regions relatively private with respect to the front door and keeping the stair easily accessible. TAC translates “relatively private” into a goal about visual openness with respect to the front door, and it translates “easily accessible” into a goal about length and change in direction along the shortest path between the stair and the exterior doors (front door and back door). TAC now has three goals: have more visual openness between the Dining and Living regions; have no more than 40% of the Dining and Living regions visible from the front door (i.e. have the visual openness values less than 0.40); have the path length and change in direction between the stair and exterior doors less than or equal to current values. Only the first of these goals, the visual openness between the Dining and Living regions, is not satisfied. To satisfy this goal, TAC proposes rotating the stair 90 degrees or 270 degrees, or moving the stair to any of six exterior edges.

5. Creating new designs

TAC carries out its suggestions, creating two new designs with the stair rotated and six new designs with the stair on an exterior edge. The two designs with rotated stair are shown below. (The other new designs are discussed in step 7.)



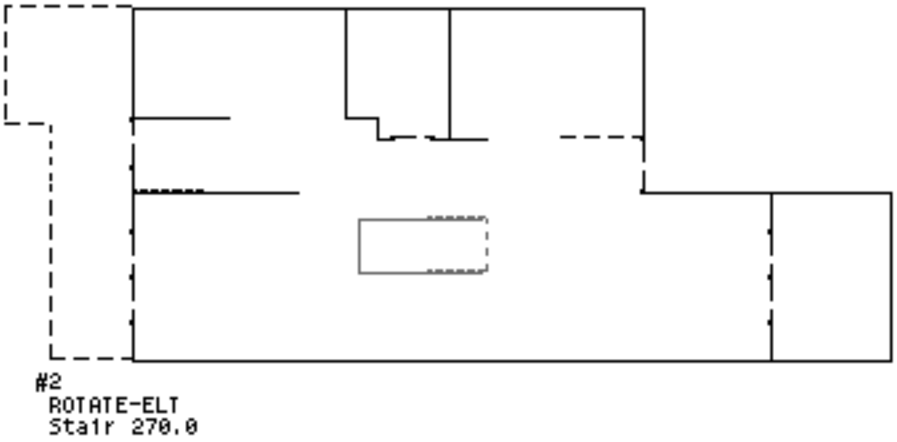
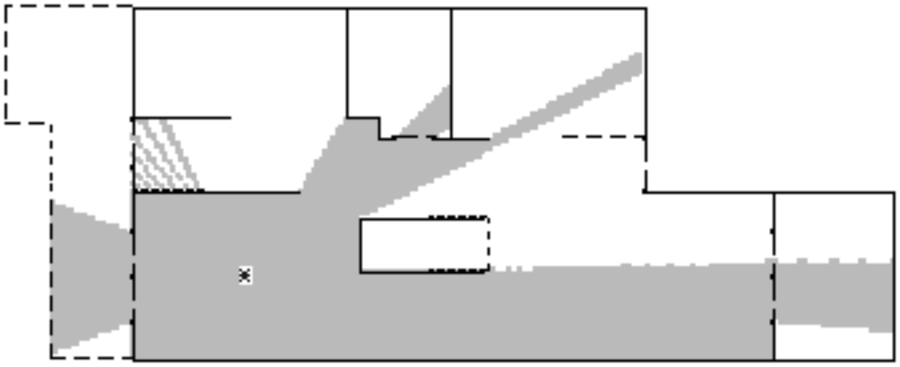


Figure 1.10: Two new designs: each has stair rotated to increase visual openness of Dining from Living.

Checking the visual openness of the Dining region in the new designs, TAC finds that more of the Dining region is now visible from the Living region so turning the stair had the intended effect. Figure 1.11 shows the visual openness measurements for one of the new designs, design #2, which is shown at the bottom of Figure 1.10.



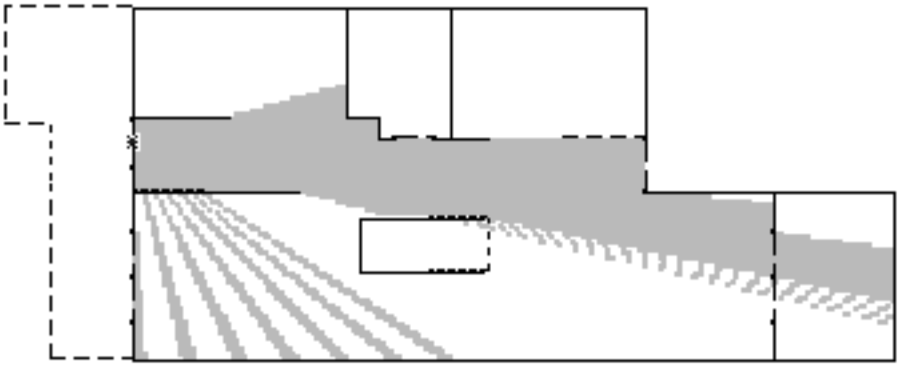
```
Visual openness from center of Living: .42
Visual openness of Dining from Living: .55
```

Figure 1.11: Visible areas and visual openness measures for design #2.

6. Evaluating all goals

The visual openness goal is satisfied for both new designs, so TAC checks that the other two goals, which were previously satisfied, are still satisfied in the designs.

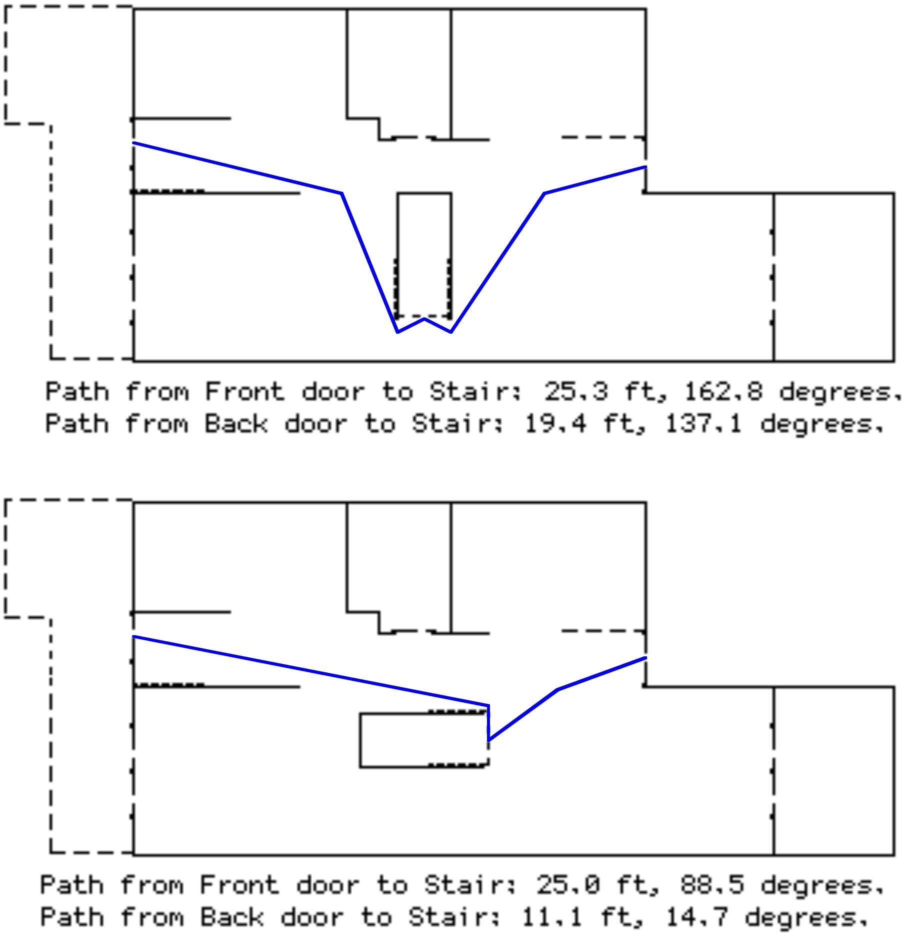
Figure 1.12 shows one of the results of checking the second goal, which specifies that the visual openness of the Living and Dining region from the front door be less than 0.40. The portion of the Living region that is visible has not changed, so it is still acceptable. A larger portion of the Dining region is visible, but it is acceptable because it is less than the specified threshold. Thus, the second goal is still satisfied.



```
Visual openness from Front-door: .32
Visual openness of Dining from Front-door: .34
Visual openness of Living from Front-door: .27
```

Figure 1.12: Visible areas and visual openness measures for front door in design #2.

Figure 1.13 shows one of the results of checking the third goal, which specifies that the stair be as "easily accessible" as in the starting design, i.e. that the values for path length and change of direction of the paths from the exterior doors be no larger than original values. In both new designs, the values have decreased.



```
Path from Front door to Stair: 25.3 ft, 162.8 degrees.
Path from Back door to Stair: 19.4 ft, 137.1 degrees.
```

```
Path from Front door to Stair: 25.0 ft, 88.5 degrees.
Path from Back door to Stair: 11.1 ft, 14.7 degrees.
```

Figure 1.13: Comparison of stair access path lengths and changes in direction for starting design (top) and new design #2 (bottom).

All three goals—to increase visual openness between the Living and Dining regions to more than the current values; to keep the visual openness of the Living and Dining regions with respect to the front door less than 0.40 (i.e. so that less than 40% of each region is visible from the front door), and to keep the path length or change in direction between the stair and exterior doors less than or equal to original values—are satisfied for both new designs, so they are solutions.

7. Displaying rejected designs

The designer asks TAC to display new designs that were rejected. The designs, shown in Figure 1.14, have the stair moved to an exterior edge. The move increased the visual openness of the Dining region from the Living region. TAC rejected these designs, however, because they do not satisfy the other two criteria, namely to keep the Dining and Living regions relatively private with respect to the front door (by keeping the visual openness values less than 0.40), and to keep the stair easily accessible from the exterior doors (by keeping path length and changes in direction less than or equal to original values). The rejected designs have more of the Dining region visible from the front door, and have at least one stair access path that is either longer or has a larger change in direction than in the starting design.

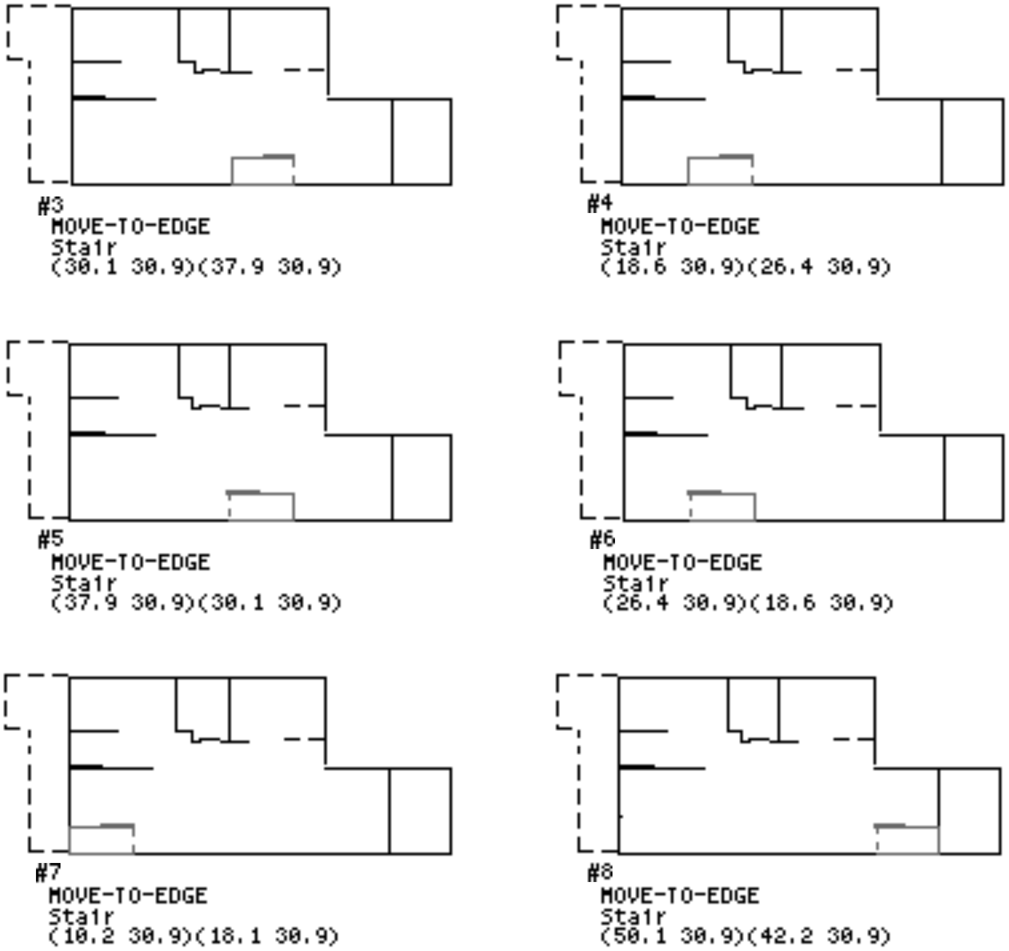

#3
MOVE-TO-EDGE
Stair
(30.1 30.9)(37.9 30.9)
#4
MOVE-TO-EDGE
Stair
(18.6 30.9)(26.4 30.9)
#5
MOVE-TO-EDGE
Stair
(37.9 30.9)(30.1 30.9)
#6
MOVE-TO-EDGE
Stair
(26.4 30.9)(18.6 30.9)
#7
MOVE-TO-EDGE
Stair
(10.2 30.9)(18.1 30.9)
#8
MOVE-TO-EDGE
Stair
(50.1 30.9)(42.2 30.9)

Figure 1.14: Rejected new designs with the stair on an exterior edge.[5]

---

[5] When moving the stair in this example, TAC did not take into account other possible stair-related issues, e.g. access to the basement and second floor.

The above example gives a flavor of how TAC works. As will be shown in subsequent chapters, TAC's languages and representations enable it to translate design goals stated in terms of experiential qualities into operators on physical form. Its control structure, which employs what we've called dependency-directed redesign, enables it to methodically and efficiently explore a large design space in search of solutions satisfying multiple goals.

TAC is notable as an artificial intelligence tool because of its knowledge, languages, representations, and dependency-directed redesign mechanism. It is notable as an architecture tool because it contributes to the clarification of terms used in architectural discourse, serves as a repository for reusable design knowledge, functions as both a design brainstorming tool and an analysis tool, and provides an example of how to distribute tasks between a designer and a computer assistant.

1.4 Motivation

TAC supports the early conceptual stage of design. Its domain is architecture, and its goal is to facilitate the design of physical form that manifests desired experiential qualities. Why focus on conceptual design? Why focus on architectural design and the relationship between physical form and human experience? Why is this problem interesting from an artificial intelligence point of view or from a designer's point of view?

Why focus on a tool for conceptual design?

Because there are very few tools for conceptual design.

In early stages of design, the language used is often very abstract. Architects and their clients use experiential terms such as "private" and "open". Engineers might talk about designing a piece of equipment that is "easy to maintain". Clothing designers talk of "baggy" clothing.

In addition, the specification of a design problem evolves along with solutions to the problem. It is rarely possible to specify apriori all design objectives and their relative priorities; some objectives may become apparent only after proposing and evaluating potential solutions (e.g. see Lawson, 1990; Akin, 1986; Schön, 1983). The designer simultaneously explores both the space of design problems and the space of design solutions.

In later stages, a designer performs more routine tasks such as resizing design elements, choosing particular design element types (e.g. kind of window) from a catalog, and positioning design elements in a well-understood portion of the design (e.g. placing refrigerator and stove in a kitchen). Computer-aided design (CAD) tools are available for these tasks. Most are drawing tools and are helpful in later stages, but require more precision than is available or appropriate for early stages of design; they are awkward to use when a design problem is not yet well-defined. In addition, there is a danger that the precision required by these tools may cause an early drawing to be perceived as a design solution.

Other computer-aided design tools are intended to be analysis tools, but are often awkward and distracting to use. Current computer-aided design systems are inadequate as design tools, and particularly as conceptual design tools.

Krauss (2000):

> Presently, designers do not use computer-aided design systems in early stages of design largely because the present generation of design programs handle only a few, generally simple sets of variables and take too much effort to use for the value gained. For example, a common type of program is one that develops a space allocation diagram, derived from a designated set of spaces and a matrix that indicates the relative importance for proximity between each set of spaces. The list of spaces usually has to be constructed specifically for the routine, and the development of the matrix, which requires cramming a set of complex judgments into a crude and laborious format, is a step extraneous to the real design process. Hence, such programs are not in popular use.

Flemming(1994):

> Frustrations generated by working with commercial CAD systems that appeared simply too “dumb” to be of use in the most interesting phases of design provided a motivation [for becoming interested in AI]. In an attempt to be applicable over the broadest range of disciplines and applications, these systems restricted themselves (and still do) to the support of the most external manifestations of design processes, the preparation of drawings and geometric models, and to the purely syntactic aspects of these representations. They did not support synthesis and interpretation, processes that are highly exploratory in nature and demand fast response and active participation from a computer system.

Why focus on architectural design and relating physical form to human experience?

Architectural design is well-suited to our research for several reasons. Most design problems exhibit the difficulties outlined above—they are exploratory in nature; involve the use of abstract, experiential terms; and require construction of both problem specification and solution. The design problems generally involve many conflicting design goals—a good test for our system. In addition, architectural theorists have written extensively about various aspects of the design process, providing a good groundwork for computational studies.

Why focus on the relationship between physical form and human experience? Because it is essential to architectural design and has not been addressed to date by computer-aided architectural design tools. Architects design spaces that people inhabit, focusing on both the physical form and how that form will affect the inhabitants. They and their clients describe spaces as private, sunny, open, spacious, inviting, etc. Architects use their knowledge from past experiences, from environment behavior research, and from their own theories to create spaces with experiential qualities such as these.[6] As discussed in the chapters that follow, this knowledge can be articulated and structured as general design principles (e.g. Wright, 1954; Moore, et al., 1974; Alexander, et al., 1977; Zeisel and Welch, 1981; Hertzberger, 1993) and used as a basis for a design support system that reasons about experiential qualities and physical form as in the opening scenario.

Why is this problem interesting from an AI point of view?

Design is inherently knowledge-based, intelligent behavior. It generally requires some degree of purposeful behavior—a designer has some number of things in mind that he wants to accomplish, then his task is to figure out how to build something that accomplishes them (while the description of what to accomplish constantly changes). When those things are experiential qualities, he must accomplish them indirectly through his choice of physical form. Classic AI issues have to be addressed in order to build computational tools to support this task. For example:

- Knowledge representation: How can abstract concepts such as “open” and “private” be operationalized? How can terms for such qualities be mapped to methods for accomplishing them? Can the same representations be used by the various components of a design tool?
- Reasoning and problem solving strategies: How can the search for design solutions avoid the combinatoric problems inherently associated with large search spaces? How are solutions found when evaluation functions are noninvertible and operators interact in complex ways? How are solutions found for multiple conflicting goals?

---

[6] For examples of the relationship between experiential qualities and physical form see the journal *Environment and Behavior*. Also see Broadbent (1980) for a good survey of environmental psychology and a discussion of the intersection of psychology, sociology, anthropology, and architecture. See Hillier and Hanson (1984) for discussion of the relationship between sociology and architecture. See Rapoport (1969, 1977) for discussion of the relationship between anthropology and architecture.

Why is this problem of interest from a designer's point of view?

Clarification of terms in architecture is a long-standing need.

> John Louis Petit (1854)
>
> I am far from thinking that nomenclature is a remedy for every defect in art or science: still I cannot but feel that confusion of terms generally springs from, and always leads to, confusion of ideas.

By providing a language and a framework for representing the relationships between experiential qualities and physical form, TAC has the potential to help clarify architectural terms. Such clarification will enable better communication among designers and between designers and clients.

In addition, TAC can serve as an example of a design tool that alleviates some of the frustrations with existing computer-aided design tools. The comments from Krauss (1970) are still apropos:

> A number of computer operations such as cost checking, sorting of program issues, and space allocation have been developed to aid the architect in his work. None, however, have been put into common usage. One of the difficulties in adopting these techniques is that they are developed for solving problems in a far different context than for architectural design. In some cases the aiding routine may require definition of design criteria which is not possible to give in the way required; in others the procedure may be so awkward to the designer that little or nothing is gained; or the tools are not sufficiently useful by themselves, without a surrounding system of more useful computer aid to warrant use.

TAC aims to be a component in such a "surrounding system", providing an intelligent partner that facilitates design by providing a rich framework for representing designs and design knowledge, and by efficiently searching for solutions.

1.5 Guide to This Document

This document is organized around the steps involved in using TAC to find solutions to a design problem. Each step is illustrated using examples taken from the set of Frank Lloyd Wright's Prairie houses. A reader interested in what TAC does, rather than how TAC works, may want to focus on Chapters 1 and 8.

Chapter 1 introduces the problem and TAC's solution, gives an example of TAC's functionality, and discusses the motivation for building TAC.

Chapter 2 describes our approach to building TAC.

Chapter 3 describes TAC's representations for design problems, designs, characteristics, and goals.

Chapter 4 describes TAC's goal evaluation.

Chapter 5 describes how TAC proposes repair suggestions using explanations.

Chapter 6 describes how TAC carries out repair suggestions to create new designs.

Chapter 7 describes TAC's control structure and discusses related issues of goal interaction, goal order, and termination. It also presents the results of experiments run to test three different control structures.

Chapter 8 describes two exercises given TAC: a design example using a local house that was in the process of being redesigned, and an analysis example using Frank Lloyd Wright's Prairie houses.

Chapter 9 discusses related work.

Chapter 10 summarizes TAC's contributions and discusses future work.

Chapter 2

Approach

This chapter presents our view of the design process and our rationale for building a design assistant instead of an automatic design generation system. It also gives an overview of TAC's dependency-directed redesign strategy and discusses the sources of TAC's intelligence.

2.1 The Design Process

TAC adopts the view that architectural design is an exploratory search of a design space, trying to turn goals, often unarticulable at the beginning of the process, into physical form that realizes those goals.[1] Several factors contribute to the need for exploration of the search space, e.g. the complexity of simultaneously satisfying many potentially conflicting goals, the fact that the goals themselves evolve along with the design, and the absence of a fixed set of modification operators that result in new designs. As a result, it is very difficult to produce feasible solutions in one or a few steps. Instead, a designer engages in an iterative cycle of goal specification and design refinement: articulate design goals, produce a potential solution, evaluate it, modify it, and continue until a solution is found. More specifically, a designer proposes an initial design for a problem, perhaps sketching a new design or selecting a previous design that solved a similar problem. He then may modify the design in an attempt to satisfy articulated goals, employing a generate-and-test model. Alternatively, he may proceed in a more opportunistic fashion, noticing new goals as he works and modifying a design in accordance with those goals rather than focusing on previously articulated goals. In either of these situations, the designer is exploring a design space via an iterative evaluation and modification cycle that is guided by characteristics he desires in a solution and knowledge about how to achieve those characteristics.

---

[1] Recorded observations of designers in action have led to this view of the design process. Rather than reiterate the observations here, the reader is referred to an in-depth study by Krauss and Meyer (1970). See Akin (1986) for a summary and reference to earlier studies (e.g. Eastman, 1969, 1970; Foz, 1973). See also Lawson (1994), Rowe (1987). See Schön (1983) for a competing theory that says designers "know it when they see it" and often cannot articulate goals. For additional theoretical views on the nature of the design process see Simon (1969, 1973, 1975), Jones (1970), Broadbent (1973), Wade (1977), Heath (1984), Cross (1991).

2.2 A Design Assistant

TAC functions as a design assistant because, even though much progress has been made in modeling design decision-making (e.g. Akin, 1986; Smithers, 1996), the design process is not yet understood well enough to build a fully automated system. We believe that an appropriate conceptual design tool acts as an assistant, complementing a human designer so that human and computer together work better, faster, and more easily than either working alone.

A human designer knows how to ascertain client needs, specify design goals, and adjust design goal sets and priorities, especially when clients' needs change; he is good at defining and extending a problem specification as he designs (e.g. Schön, 1983; Akin, 1986; Rowe, 1987; Lawson, 1990). He is good at generating and sketching initial designs, before design goals can be articulated. He knows how to shift his focus of attention between various aspects of a design. He knows how to dynamically add to his design knowledge, inventing new ways to satisfy design goals as he designs. He also knows how to judge when to stop, which is often not obvious because there is no single right answer.[2]

Computers have a valuable role to play as assistants. They can explore a design space methodically and can manage the complexity of many conflicting design goals. Humans have trouble dealing with such complexity due to limited capacity of short-term memory and the difficulty of deducing all consequences of particular design modifications. Computers are far less susceptible to such overload. Computers can provide languages and frameworks for helping a designer articulate and store design knowledge. They also can more easily and quickly generate and keep track of alternatives, and modify and redraw representations of designs.

TAC takes on the tasks computers do well, leaving the tasks humans do well to the designer.

2.3 TAC as Design Assistant

As noted earlier, current CAD tools do not support exploration, and as a result are often extraneous to the design process. They also have simple design vocabularies and require too much precision. TAC assists the designer with all but one of these problems. TAC supports exploration by focusing on the evaluation and repair cycle that is central to design, thereby being an integral part of the process rather than extraneous to it. It assists a designer in specifying design goals, evaluating a design with respect to those goals, and repairing the design if necessary. In the process, it generates alternatives easily and quickly, helping to manage the complexity of satisfying multiple goals in a domain in which modification operators interact and evaluation functions are not easily inverted. In addition, its broad vocabulary is that of the designer, and as such, includes terms for experiential qualities and elements of physical form. TAC does not deal with the precision issue that hampers CAD drawing tools.

---

[2] See discussion of satisficing in Simon (1969).

There are aspects of the design process that TAC does not address. It leaves the task of generating an initial design to the designer. It also leaves the task of design goal specification and respecification to the designer, focusing on design refinement rather than goal refinement. Finally, it generates multiple solutions when they exist, leaving the task of ranking the solutions to the designer.

2.3.1 Dependency-Directed Redesign

TAC's dependency-directed redesign strategy embodies a variation of generate-and-test (or more accurately, test-and-generate): given an initial design and a set of goals, TAC evaluates a design with respect to the goals, then generates new designs by proposing, refining, and carrying out repair suggestions (i.e. design modifications). The generate step, which we call repair, is dependency-directed, guided by knowledge of two kinds of dependencies: general dependencies between experiential qualities and physical form, which include methods for realizing qualities in form; and specific dependencies for a particular design, which serve as explanations for unsatisfied goals. If a design does not satisfy all goals, TAC uses the explanations to search its knowledge base for design modification operators likely to repair the design so that all goals become satisfied.

Three aspects of the architecture domain motivated our choice of dependency-directed redesign as a control structure: some evaluation functions cannot be easily inverted to set the values of design characteristics, the effects of operators are difficult to predict, and multiple interacting goals cannot be easily satisfied in one or a few steps.

1. The values of some design characteristics cannot be set.

Some design modification operators directly set the values of design characteristics. Changing the color of a door in a design, for example, is straightforward because color is a directly settable design attribute. Many design characteristics in this domain—and all that represent experiential qualities—are not directly settable, however. Instead, their values are changed by modifying the physical form. If we want half of the living room visible from the dining room, for example, we might turn an intervening stair as in the opening scenario. More of the living room may indeed be visible, but it is difficult to guarantee that the visibility value will be 0.5. Visibility is an example of a characteristic measured via a computational geometry routine whose result depends crucially on the specific arrangement of design elements, and that is therefore not easily inverted. For such a characteristic it is easier to carry out an operator and measure the characteristic in the resulting new design, i.e. to employ TAC's version of generate and test.

2. The effects of operators are difficult to predict.

Effects of operators are difficult to predict because an operator may not have its intended effect or may have unintended effects, and because operators may interact.

An operator has an intended effect that is stated in terms of a desired value for a characteristic of a design, e.g. increasing the number of fireplaces in a design. When design characteristics can be measured independently of context, as with counting the number of fireplaces, the effects of operators that modify those characteristics can be predicted. Adding a fireplace always increases the number of fireplaces in a design regardless of the specific geometry of the design. Other characteristics depend on the particular arrangement of design elements, and the effects of operators intended to achieve those characteristics cannot be predicted. To make all of one space visible from another, for example, a wall might be moved. For some arrangements of design elements, moving the wall may satisfy the visibility goal. In others, an intervening design element (e.g. a stair) may keep the move operator from having its intended effect on visibility. Attempting to predict whether the goal will be satisfied requires just as much work as performing the move operation and checking the design.

An operator also may have unintended effects. Moving the wall may increase visibility between two spaces as intended, but may block the view between two other spaces. It is very difficult at best, and impossible at worst, to precompute all effects of operators; it is easier to carry out an operator, then check the design to determine which aspects of the design have been affected.

Finally, the effects of combinations of operators can be more difficult to predict than effects of single operators. Operators can interact in complicated ways: one operator may move a fireplace to a particular location, displacing a bookcase; a second operator may move the bookcase back to its original location, moving the fireplace to yet another location; a third operator may move the bookcase yet again. Because each move causes the design to be slightly different, new locations may become apparent after performing the move. As a result, it is difficult to predict where each design element will end up, much less what the effect will be on characteristics such as visibility. It is easier to determine the combined effect of a sequence of operators by carrying out the operators.

Each of these examples has illustrated that a generate and test strategy is appropriate when the effects of operators are unpredictable.

3.Multiple interacting goals cannot be easily satisfied in one or a few steps.

As previously mentioned, there is no obvious path from problem definition to solution, especially with interacting design goals and operators whose effects are not completely predictable. It is possible, however, to generate a design that satisfies a subset of the design goals. Such designs, even if not solutions, are often useful intermediate steps from which progress toward a solution can be made. An enabling assumption for this approach is that some goals will be independent, so that working on one goal does not always cause a previously satisfied goal to become unsatisfied. For the goals that do interact, some amount of work to reevaluate and resatisfy goals is necessary, hence the iteration between evaluation and repair.

The difficulties of reasoning with noninvertible evaluation functions, predicting operator effects, and of satisfying multiple design goals in this domain lead to the choice of an iterative control structure that focuses on evaluation and repair. TAC proposes and refines suggestions for modifying a design, then carries out the suggestions and evaluates the resulting new designs to see if intended goals are satisfied. If some of the goals are now satisfied for a new design, TAC assumes that it is making progress and attempts to repair the design by starting a new evaluation and repair cycle.

2.3.2 Intelligence

TAC's intelligence derives from the richness of its knowledge base and representations, and its informed search of repair suggestion space.

Knowledge

TAC's knowledge consists of both general knowledge and domain knowledge. The general knowledge is used by TAC's reasoners to explain their reasoning, do simple arithmetic, compare and rank vectors using a partial order, deduce part-whole relationships, and manipulate nodes and arcs in a graph.

The domain knowledge is a major source of TAC's power as an architectural design assistant. It includes general design principles as well as specific methods for modifying designs. It is organized around types of design elements, which represent physical form such as walls and doors; and design characteristics, which represent properties of a design and range from very concrete details such as square footage to very abstract qualities such as openness or privacy. Much of TAC's intelligence derives from its mapping of experiential qualities to details of physical form. TAC knows, for example, that it can make one space more visible from another by removing intervening walls. It knows that it can make a space feel more private by making less of it visible or making the path to it from a front door less direct.

TAC also knows about characteristics of Frank Lloyd Wright's Prairie houses. It knows, for example, that Prairie houses often have large main living spaces containing a fireplace that symbolizes home and hearth. (See Section 8.2 for discussion.) Such knowledge serves to illustrate that TAC can represent not only general architectural knowledge, but a designer's particular preferences as well.

Representation

TAC's representation languages enable close coupling between TAC's components. The results of one component are used as input for the next: evaluation results in the form of explanations are the input to the repair suggestion mechanism; repair suggestions are the input to design creation. TAC's languages enable this smooth integration by sharing a common syntax and by being expressive, concise, and unambiguous.

TAC's representation of designs supports evaluation of a wide variety of design characteristics. Some of the characteristics are concerned with the geometry of design elements (e.g their sizes and location); others are concerned with the boundaries of spaces or circulation paths in a design. TAC handles the requirements for different information by representing a design as a set of models, each representing a different aspect of a design. It then chooses the model appropriate for a particular task.

Reasoning and Search

Exhaustive search of the design space is not an option in this domain; there are far too many possible designs. TAC limits its search by first searching repair suggestion space and pruning that space, so that it proposes only repairs that have a good chance of leading to solutions. It carries out the repair suggestions, modifying the original design to create new designs that it thinks are likely to satisfy at least some of the specified goals. It then attempts to repair any of the new designs that are not solutions.

TAC's repair suggestion mechanism identifies relevant suggestions by comparing desired values of design characteristics with actual values. It figures out, for example, that if a characteristic's actual value is less than desired, it must increase the value. It checks its knowledge base for modifications that are likely to increase the characteristic's value, then checks the particular design to determine which of the modifications are relevant to the current context. It proposes and carries out the selected modifications, then checks each resulting new design to make sure the modification had the intended effect.

In summary, TAC is an intelligent design assistant that employs a strategy we've called dependency-directed redesign which enables it to evaluate designs with respect to a set of design goals, propose and refine repair suggestions, then carry out those suggestions to create new designs that satisfy the goals.

Chapter 3

Defining a Design Problem

TAC is organized around the notion of a design problem, by which we mean a design and a set of design goals. Let's say that we have a design and a set of design goals of the sort mentioned in the opening scenario, e.g. we want living spaces that feel “open” to one another, but “private” with respect to the front door, and we want the stair to be “easily accessible”. We also might want the design to have four bedrooms. It's fairly easy to translate having four bedrooms into a particular design; it’s much harder to translate qualities such as open, private, and accessible. The mapping between these qualities and physical form is not straightforward. It is possible, however, to represent these qualities in such a way that they can be measured and reasoned about, and thus used to create designs that satisfy desired goals.

3.1 Representing Designs

A design is represented by five kinds of models—a design element model, an edge model, a territory model, a use space model, and a circulation model. Each of the models abstracts different design details, enabling easier, faster access to information relevant to a particular task.

The *design element model* contains information about design elements, i.e. the objects that create physical form: walls, windows, doorways, fireplaces, etc. Included is information about size, location, materials, and component parts.

The *edge model* is a two-dimensional geometric abstraction of the design element model and contains points and nonoverlapping edges. Each edge in the edge model is the result of applying one or more abstraction methods to particular design elements. The information summarizing the origins of an edge is called the edge's derivation, and as described later, is used by TAC when suggesting repairs and creating new designs. There are three edge derivation methods: an edge may be a one-dimensional abstraction of a design element, such as a wall; a member of a two-dimensional footprint (i.e. outline) of a design element, such as a fireplace; or a one-dimensional projection of a design element. A projection, also called a projected edge, does not correspond to a physical object, but is an extension of edges derived from design elements. As shown in Figure 3.1, projected edges extend in either parallel or orthogonal directions from a design element.