Artificial Intelligence for Engineering Design, Analysis and Manufacturing (1998), *12,* 141–159. Printed in the USA.

SPIDA: Abstracting and generalizing layout design cases

D. MANFAAT,[1,2] A.H.B. DUFFY,[1] AND B.S. LEE[2]

[1]CAD Centre, Department of Design, Manufacture and Engineering Management, University of Strathclyde, 75 Montrose Street, Glasgow G1 1XJ, Scotland, UK

[2]Department of Ship and Marine Technology, University of Strathclyde, 100 Montrose Street, Glasgow G4 0LZ, Scotland, UK

(Received June 27, 1997; Revised October 17, 1997; Accepted November 10, 1997)

Abstract

Abstraction and generalization of layout design cases generate new knowledge that is more widely applicable to use than specific design cases. The abstraction and generalization of design cases into hierarchical levels of abstractions provide the designer with the flexibility to apply any level of abstract and generalized knowledge for a new layout design problem. Existing case-based layout learning (CBLL) systems abstract and generalize cases into single levels of abstractions, but not into a hierarchy. In this paper, we propose a new approach, termed *customized viewpoint—spatial* (CV–S), which supports the generalization and abstraction of spatial layouts into hierarchies along with a supporting system, SPIDA (SPatial Intelligent Design Assistant).

Keywords: Abstraction; Generalization; Machine Learning; Pattern Matching; Spatial Layout Design

1. INTRODUCTION

The utilization of spatial layout design experience can be considered as being a key part of the spatial layout design process in that a form of design experience is used or modified to produce a new design solution (Foz, 1973; Akin, 1978; Jones, 1980). Here, design experience refers to *specific* past spatial layout design cases and *abstract* and *generalized* knowledge generated from these cases by the processes of abstraction and generalization. Abstraction can be considered as being a process of reducing the complexity of an object (Simon, 1981; Darden, 1987; Coyne & Flemming, 1990; Oxman, 1990; Hoover et al., 1991). Generalization is defined as being an inference process or learning from examples, aiming to generate concepts whose descriptions are more general than those of the examples (Cohen & Feigenbaum, 1982). Thus, a basic distinction made in this paper is that abstraction reduces detail but can be applied to a single case, whereas generalization generates new knowledge from more than one case and thus represents more widely applicable knowledge.

Generalization of past spatial layout design cases into *hierarchical levels* of abstractions provides the designer with the flexibility to apply any level of abstract and generalized knowledge to a new design problem (Duffy, 1993; Duffy & Kerr, 1993). It also supports an efficient search for the best matched cases because it provides a reduced search space of the cases (Oxman, 1990). That is, the search is directed from the higher to lower levels of abstract and generalized knowledge in the hierarchy rather than to the cases. Similarly, the abstraction of a single case into *hierarchical levels* of abstractions provides the designer with the flexibility to choose any level of abstraction that he or she needs for generating a new design solution.

In the field of spatial design, existing case-based layout learning (CBLL) systems support the abstraction and generalization of layout design cases by learning from the cases (Coyne et al., 1989; Coyne & Postmus, 1990). However, they *do not* generalize design cases into a hiearchy, but rather generalize the cases into a single abstract level. Consequently, the designer is restricted to using the abstract and generalized knowledge of only one level of abstraction of the cases. In addition, when searching for the best matched cases, these systems can only reduce the search space by considering a single level of abstraction. Thus, guided searching through abstract hierarchies in a top-down manner is prohibited. The systems also *do not* abstract a single case into a hierarchy according to the designer's needs.

In this paper, we present a new approach, called the *customized viewpoint—spatial* (CV–S), which is proposed to

Reprint requests to: D. Manfaat, CAD Centre, Dept. of Design, Manufacture and Engineering Management, University of Strathclyde, 75 Montrose Street, Glasgow G1 1XJ, Scotland, UK. Tel: (+44) 141-552-4400; Fax: (+44) 141-552-3148; E-mail: joe@cad.strath.ac.uk

overcome the above limitations that existing CBLL systems have. It builds on the work of Duffy and Kerr, who advocate that viewpoints should be customized to suit particular designers' needs (Duffy & Kerr, 1993; Kerr, 1993). Thus, the CV–S approach supports the effective utilization of spatial layout design experience by generalizing past spatial layout design cases and abstracting a single case into hierarchical levels of abstractions according to the designer's needs. In addition, the realization of this approach in a computer system is presented.

To introduce the CV–S approach and its realization in a computer system, the existing CBLL systems are briefly reviewed in Section 2. Aspects of pattern matching as a key process of generalization of layouts are discussed briefly in Section 3. The CV–S approach is presented in Section 4, followed by the key descriptions of techniques used to implement the approach in Section 5. The implementation of the CV–S approach in a computer system, SPIDA, is presented in Section 6. An evaluation of the SPIDA system in the light of dimensions of learning in design as defined by Grecu and Brown (1996) is presented in Section 7. Finally, a conclusion of the work presented in this paper is presented in Section 8.

2. EXISTING CBLL SYSTEMS

Systems developed by Coyne et al. (1989) and Coyne and Postmus (1990) are examples of the existing CBLL systems. Both systems use neural networks to represent and generalize past spatial layout design cases. In general, the systems learn a set of patterns of design cases and acquire general descriptions of the patterns. When presented with a partial input pattern, the systems produce the output pattern by completing the input pattern, using the acquired general descriptions. Consequently, the result of this generalization process is a single-level abstraction of the design cases in the form of the output pattern.

In the above systems, pattern matching is involved when matching a partial input pattern against a set of learned patterns, to produce an output pattern that represents the general descriptions of the learned patterns. Thus, pattern matching is a key process of the generalization process and one that has received little attention in spatial layout design.

3. OVERVIEW OF PATTERN MATCHING

Pattern matching can be defined simply as the activity of matching patterns with the aim of finding similarities between them for the purpose of recognition and/or retrieval of similar patterns. One field where pattern matching is involved is pattern recognition (Fu, 1976). In this field, pattern matching is used to decide how closely an input pattern "fits" a class of patterns, thus classifying the input pattern into a predetermined class. It also plays a vital role in design, where the process often is greatly assisted by identifying, retrieving, and then modifying appropriate past design cases, and generalizing the cases, with the implied benefit of being able to utilize relevant past experience (Maher & Zhao, 1987; Duffy & Kerr, 1993).

A number of approaches with their diverse techniques have been developed in pattern matching. The three main aspects that characterize a pattern matching technique are *pattern classes*, *degree of similarity*, and *matching methods.* These three main aspects are addressed briefly in the following three subsections.

3.1. Pattern classes

Pattern classes refer to the classes of patterns that are used in the pattern matching process. The four basic classes of patterns are *geometric patterns*, *topological relations*, *distributed patterns*, and *semantic/symbolic patterns* (see Figure 1). Patterns in the class geometric patterns, for example, *point sets*, *dimensional graphs*, and *drawings*, are represented in coordinate systems. The class topological relations is represented as a network or a graph made up of vertices, edges, and faces. This network or graph represents connections of the elements of a pattern. Patterns in the class distributed patterns, for example, *spatial patterns*, *spatial relationship patterns*, and *graphical patterns*, are distributed across a matrix of grid units or of pixels in computer monitors. Examples of the class semantic/symbolic patterns are *text* and *symbols* (such as diagrams, icons, etc.).

3.2. Degree of similarity

The degree of similarity reflects the amount of match between patterns of matched objects. There are different approaches to defining the degree of similarity that may exist between matched objects. For example, Smith (1989) classifies the degree of similarity into five classes: *resemblances*, *overall similarity*, *identity*, *part-similarity*, and *part-identity* (see Figure 2). In Figure 2, each class is illustrated with three examples, each showing the relationship between two objects in a pair. The attributes of each object in the pair, such as the color, shape, and dimension, are used to compare the objects.

The class *resemblances* is an all-encompassing class of similarity that includes the other four classes and is, therefore, similarity at its most "undisciplined"—unconstrained and unspecified (Smith, 1989). The classes *overall similarity* and *identity* are concerned with a whole-object similarity that takes into account all of an object's characteristics at once, and no particular attributes are emphasized. Overall similarity is defined for objects that are discriminably different but also highly similar overall, while identity is concerned with objects that are the same. The classes *part-similarity* and *part-identity* are concerned with the constituent attributes of objects, that is, particular aspects or attributes are emphasized in the comparison. *Part-similarity* is defined for objects whose particular attributes are considered similar, while *part-identity* is defined for objects that

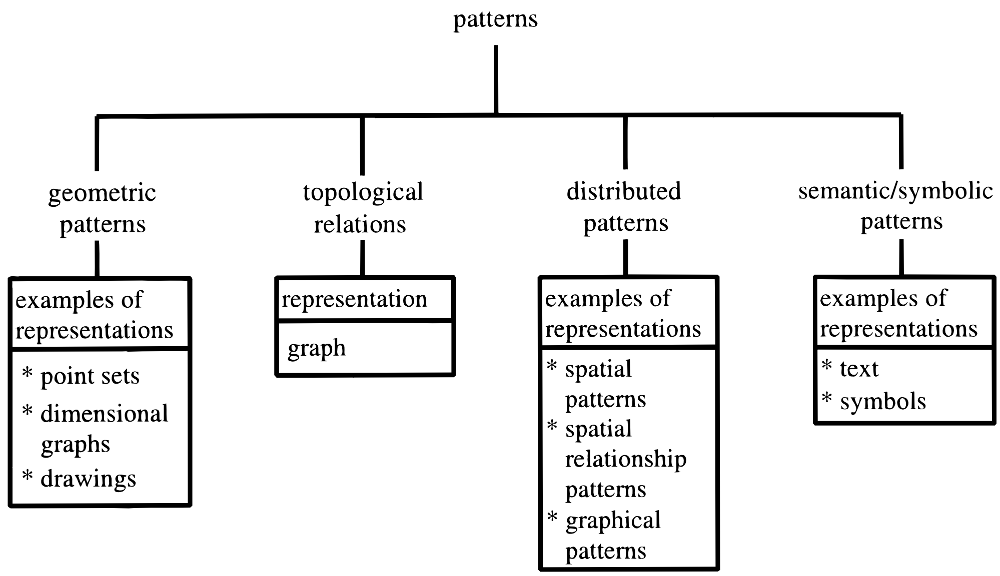


Fig. 1. Four basic classes of patterns (Manfaat et al., 1996).

Five classes of degree of similarity	Pairs of objects a	Pairs of objects b	Pairs of objects c	Similarity: Attributes	a	b	c
1. Resemblances --- sets of objects that are "alike". That is, there is some kind of similarity present either in part or whole.				Colour	✗	✓✓	✗
				Shape	✓✓	✓✓	✓
				Size	✗	✓✓	✓
2. Overall similarity --- sets of objects similar overall. That is, more attributes are similar than dis-similar.				Colour	✓	✓✓	✓
				Shape	✓✓	✓✓	✓
				Size	✗	✓✓	✓
3. Identity --- equivalence sets of identical objects. That is, all attributes are the same/identical.				Colour	✓✓	✓✓	✓✓
				Shape	✓✓	✓✓	✓✓
				Size	✓✓	✓✓	✓✓
4. Part-similarity --- sets of objects similar in part. That is, some attributes are similar.				Colour	✗	✗	✓
				Shape	✓✓	✗	✓
				Size	✗	✓	✓
5. Part-identity --- sets of objects identical in part or sharing common particular attributes. That is, some attributes are the same/identical.				Colour	✓✓	✗	✗
				Shape	✗	✓✓	✓✓
				Size	✗	✗	✓✓

Key: ✓✓ = Same ✓ = Similar ✗ = Dis-similar

Fig. 2. Illustrations of classes of the degree of similarity (adapted from Smith, 1989).

https://doi.org/10.1017/S0890060498122060 Published online by Cambridge University Press

are identical in their particular attributes or share common particular attributes.

3.3. Matching methods

The underlying methodology on which matching processes are based can also have significant effects on the characteristics of pattern matching techniques, and thus is an important issue when considering adoption for specific applications. In design, different matching methods are used. An example is *neural networks* used in the CBLL systems mentioned in Section 2. The method is used to match a partial input pattern against a set of learned patterns of layout design cases. Another method is *symbolic pattern matching*, which is, in most cases, used in case-based design (CBD) systems (Bareiss, 1991; Kolodner, 1993) to match design attributes. Other methods are *bitmap matching* and *graph-theoretic algorithms*, such as those used in a CBD system called FABEL (Voß et al., 1994). The former is used to match images of layout design cases, while the latter are used to match topological relations of the furniture of room design cases.

4. CUSTOMIZED VIEWPOINT—SPATIAL APPROACH

In this paper, we present a new approach to abstracting and generalizing layout design cases called *customized viewpoint—spatial* (CV–S). This approach is aimed to overcome the limitations that existing CBLL systems have. It is based on the *customized viewpoint* (CV) approach (Duffy & Kerr, 1993; Kerr, 1993), which advocates the abstraction and generalization of specific experiential design knowledge into appropriate viewpoints according to the designer's needs. That is, the designer can select particular viewpoints for abstraction and generalization. The CV approach focuses on numerical design. Our CV–S approach is concerned with spatial design.

The CV–S approach has two separate parts: *generalization* of layouts and *abstraction* of a single layout into hierarchical levels of abstractions, based on different viewpoints, according to the designer's needs (see Figure 3). The generalization part focuses on layouts and their different levels of generalized knowledge, while the abstraction part focuses on a layout and its levels of abstractions. In the case of abstracting a layout, the designer can browse over the set of layouts and select a layout for the abstraction into a hierarchy. The designer then can utilize any level of a layout's abstractions in the hierarchy for a new design problem.

4.1. Generalization of spatial layouts

In the synthesis stage of *original* design, that is, design that involves elaborating original solution principles for the design problem to form a new design product (Pahl & Beitz, 1988), the design solution generated is based on previously defined design attributes. In spatial design, the designer often uses diagrams (e.g., "bubble diagrams") and sketches to help him or her generate initial layout design solutions (Jones, 1963). In "diagramming," the designer draws the diagrams of adjacencies between spaces that result in the topological pattern of the layout. While in sketching the designer draws the spaces, each with its geometric shape, based on the diagrams of spatial adjacencies, thus combining the topological pattern and geometric shape of the layout. Consequently, in a spatial layout design process that directly utilizes past layout solutions or their abstractions as the initial solution, there are two main advantages:

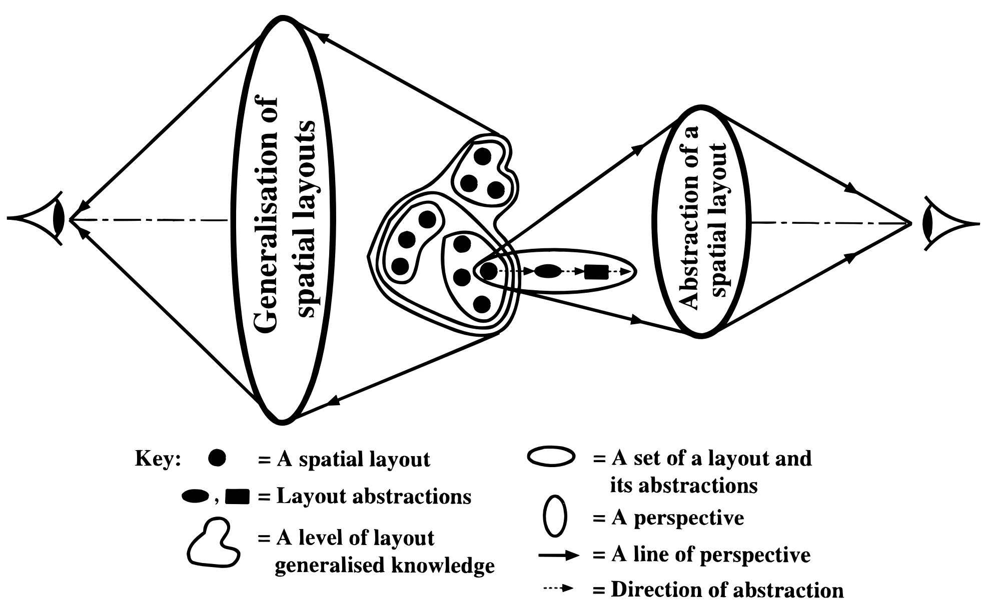


Fig. 3. CV–S approach.

https://doi.org/10.1017/S0890060498122060 Published online by Cambridge University Press

- the designer's task of generating the solution can be reduced; and
- the designer can be provided with a suitable topological pattern or combination of topological pattern and geometric layout.

As addressed in Section 2, by using neural networks, the existing CBLL systems are able to generalize patterns of layout designs, for example, spatial patterns and spatial adjacency patterns. Given a partial input pattern, the systems complete this pattern using the general descriptions acquired from a set of learned patterns. However, the drawbacks of neural networks are that a change to the input pattern (e.g., due to rotation, scaling, translation, etc.) should result in a mismatch of the output pattern (Manfaat et al., 1996), and that they are essentially "black boxes," in that the knowledge they contain is not explicit with the same clarity, for example, as symbolic representations (e.g., rules in an expert system) (Coyne et al., 1989).

Considering the drawbacks of neural networks and the advantages of using past layout solutions, we present two approaches to generalizing layouts into a hierarchy based on two different viewpoints: topology (similarities in spatial adjacencies) and a combination of topology and geometry (similarities in spatial adjacencies and geometric shapes), respectively. These two approaches, and a number of *key* issues that should be taken into account if either of these two approaches is to be pursued, will be discussed briefly in the following three sections.

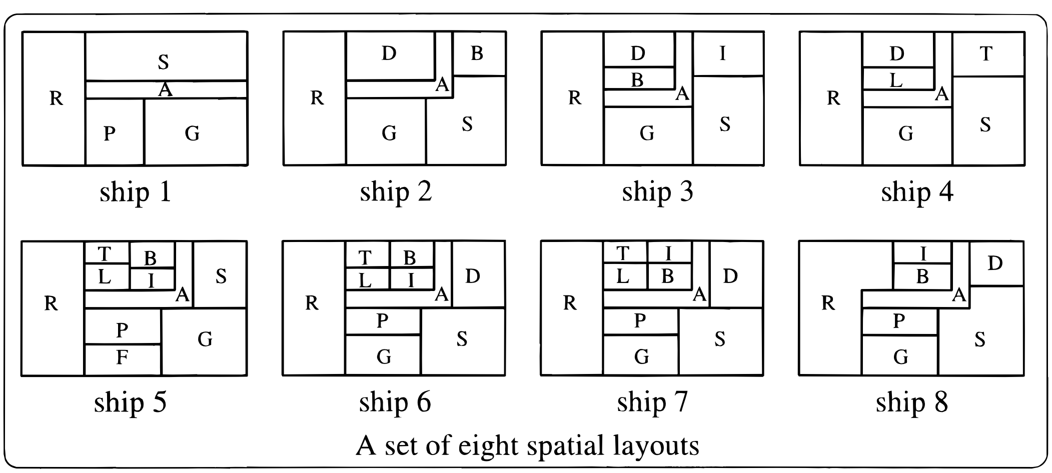
4.1.1. Topological generalization

In this paper, the word *topology* refers to adjacencies between the spaces of a layout in terms of the nature of the spaces. The name of a space expresses the nature of the space. Two spaces, for example, *galley1* and *galley2*, have the same nature, that is, a galley. In the nature of a space, there is an inherent function that allows analogies. For example, a galley and a pantry are analogous to spaces used for the preparation and cooking of food.

The topology of a layout can be represented as an adjacency graph. As an example, consider a set of eight spatial layouts of the catering decks of passenger ships given in Figure 4. When generalizing this set based on topology, the hierarchy of adjacency graphs given in Figure 5 may represent the result of this generalization. The grouping of the layouts and their abstractions, respectively, are based on similarities in these graphs, that is, the number of the same adjacency relations that preserve the corresponding nodes (spaces) between the graphs. For example, at the lowest level of the hierarchy, ships 6, 7, and 8 share the maximum number of adjacency relations that preserve the corresponding nodes, as compared to the graphs of the other ship layouts. Therefore, the graphs of these three ships are generalized into a generalized graph, *gg1*, one level above them. The process of generalization continues to the higher levels until the overall hierarchy of adjacency graph abstractions is generated.

4.1.2. Combined topological and geometric generalization

When generalizing spatial layouts based on the combination of the topological patterns and geometric shapes of the layouts, topological pattern matching between the layouts is initially carried out. This is because, as discussed previously, in the spatial design process the designer often generates the initial solution based on "bubble diagrams" and sketches. The diagrams represent patterns of connectivity (topology) between spaces. In sketching the layout the topology is also represented, in addition to other features, such as shapes, sizes, and so on. Thus, topology is one of the



Key: A = Alleyway, B = Bar, D = Disco, F = Food store, G = Galley, I = Information office, L = Lobby, P = Pantry, R = Restaurant, S = Shop, T = Toilet

Fig. 4. A set of eight spatial layouts of the catering decks of passenger ships.

https://doi.org/10.1017/S0890060498122060 Published online by Cambridge University Press

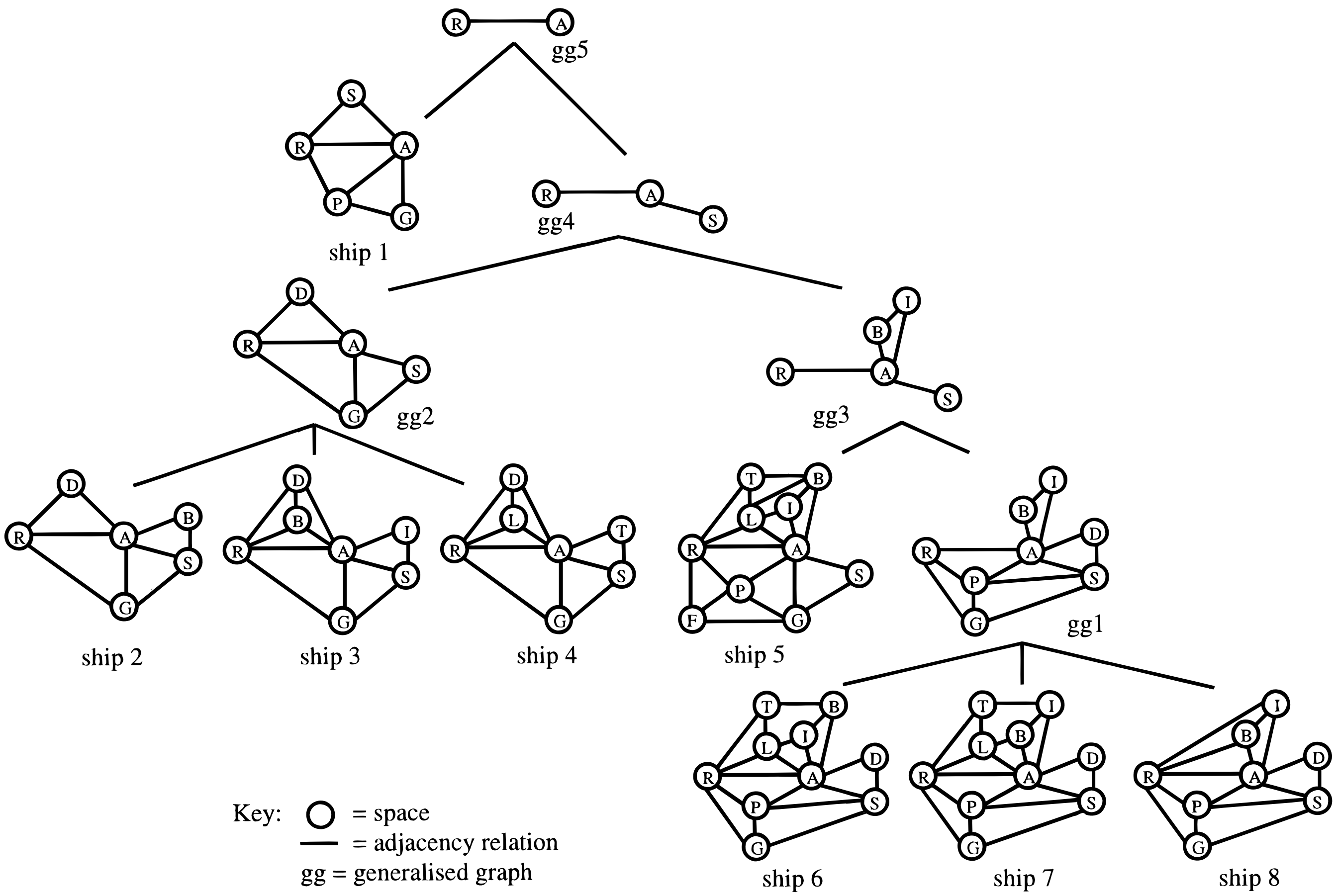


Fig. 5. Generalization of graphs of spatial adjacencies of the layouts in Figure 4.

main features of a layout that initially should be taken into account when generating the solution. Once the corresponding spaces of each of the matching layouts have been defined, geometric shape matching then is applied to each of these corresponding spaces. The result of the matching of each of these corresponding spaces gives a measure of the similarity between the spaces. Accumulating the measures of similarity of all of the corresponding spaces gives an overall measure of the similarity between the matched layouts. The clustering of the layouts and their abstractions, respectively, is based on their measures of similarity.

As an example, when generalizing the set of layouts in Figure 4 based on the combination of topological patterns and geometric shapes of layouts, we may have a hierarchy of generalized layouts as shown in Figure 6. In this figure, in terms of adjacency graphs, the layouts of ships 6, 7, and 8 are similar (see Figure 5). However, in terms of geometric shapes, the layouts of ships 6 and 7 are more similar than if each of these layouts is compared to the layout of ship 8. That is, all of the shapes in ships 6 and 7 are the same, but in ship 8 the restaurant (R) and shop (S) are different. Therefore, the layouts of ships 6 and 7 are generalized into a generalized layout, *gl1*, one level above them. The generalization process continues to the higher levels until the overall hierarchy of layout abstractions is generated.

4.1.3. Key issues

Realizing these two approaches to learning from past designs, within a computer support environment, gives rise to a number of *key* issues:

- *How to represent the layouts?* Representations of the layouts should support the processes involved, such as pattern matching, clustering, generalization, and retrieval of the layouts and their abstractions. For example, closed polygons may be used to represent the geometric shape or pattern of a layout. These polygons may be transformed into a different representation in order for a process, for example, pattern matching, to be carried out.
- *How to pattern-match the layouts?* To generalize the layouts, we need to cluster them into groups. To cluster them, we need to pattern-match one layout to another. For example, given two geometric patterns to match, the pattern-matching process should be carried out in such a way that the result of matching is as accurate as possible. Further, if there is a change to either or both of the patterns (e.g., due to rotation, scaling, translation, etc.), the accuracy of matching should not be affected. That is, the matching process should be robust.

https://doi.org/10.1017/S0890060498122060 Published online by Cambridge University Press

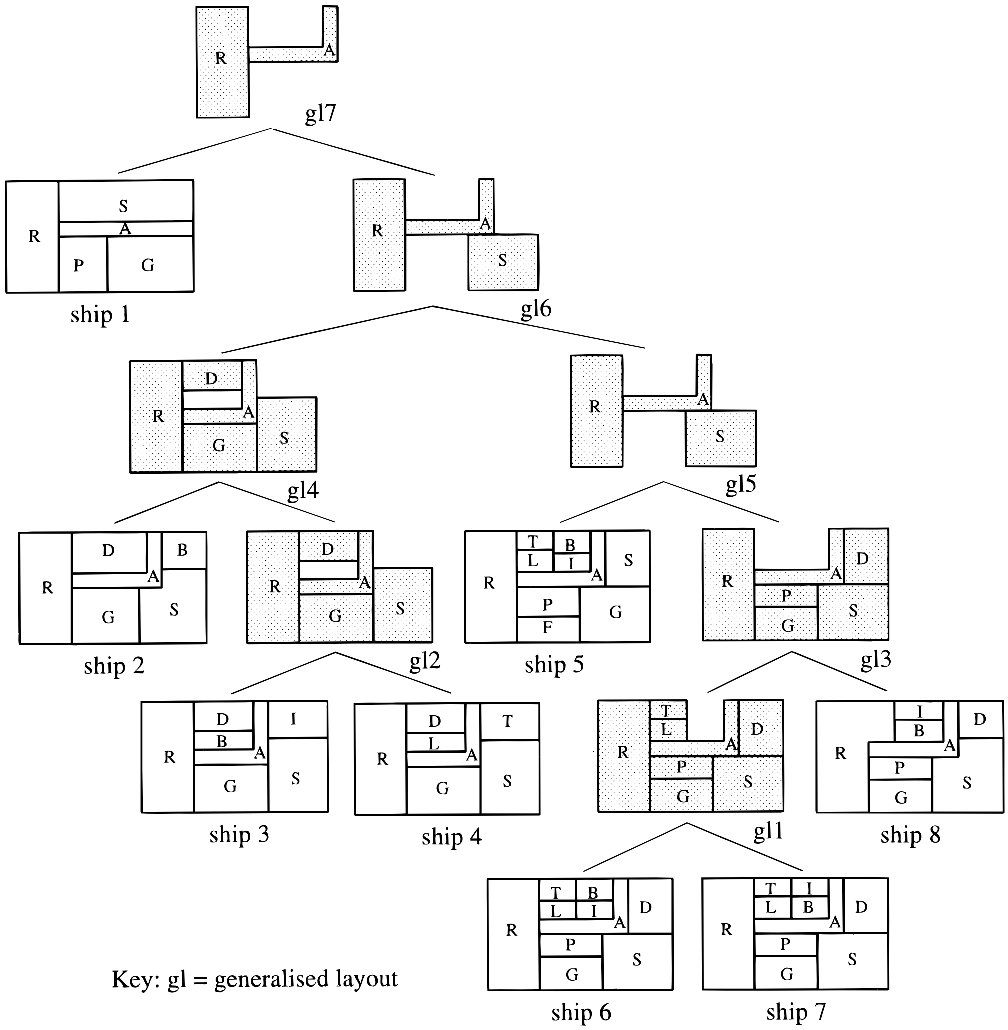


Fig. 6. Generalization of spatial layouts based on the combination of their topological patterns and geometric shapes.

- *How to measure similarities between the layouts?* Matching layouts results in a degree of similarity, such as an overall or a part similarity, between them. It is carried out by comparing the parts and/or attributes of one layout with those of another. For example, in matching two graphs representing two topological patterns of layouts, the measuring of similarities between the graphs is carried out by finding the correspondences between the nodes of one graph and those of another graph, which preserve the correspondences between the links of the graphs.
- *How to cluster similar layouts?* Based on the measures of similarities between the layouts, the layouts can be clustered into groups. The clustering should be done in such a way that the layouts that have a higher degree of similarity should be clustered in the same group. The groups of the layouts should further be able to be clustered into higher level groups representing abstractions of the layouts.
- *How to generalize the layouts into a hierarchy?* Given clusters of layouts or abstractions, it should be possible to generalize the layouts or abstractions in each cluster to generate the abstract and generalized (i.e., learned) knowledge. This leads to the generalization of layouts into a hierarchy. In each level of generalization, the generalization process is carried out by preserving the common parts and/or attributes of the layouts or abstractions and neglecting the other parts and/or attributes that are significantly different. For example, in abstracting the adjacency graphs of layouts, the corresponding nodes (spaces) that have the correspondences of adjacency relations are preserved, while the other nodes are dropped. It also should be noted that different hierarchies can be generated. This is because when matching the topological patterns of two layouts in both kinds of generalizations, for example, there can be different sets of corresponding spaces that have the same number of adjacency relations between the spaces. This means that different clusters and hence different generalizations may be generated.
- *How to search, match, and retrieve similar layouts given an input layout?* To retrieve layouts similar to a given

input layout, we need to search the layouts and their abstractions and match the input against them. The input given by the designer should be represented in such a way that these searching and matching processes can be carried out. Once the matching layouts or abstractions are found, it should be possible to retrieve the layouts and/or abstractions.

- *How to update the hierarchical abstractions of layouts when new layouts are added?* This involves augmenting the hierarchical abstractions of layouts by adding new layouts. Will the added layouts be accommodated into particular clusters of layouts/abstractions as the new instances of these clusters? Or will these hierarchical abstractions be regenerated by including the added layouts?

By addressing such issues, a resulting system will be able to learn, and make explicit, previously implicit knowledge through the generation of abstract hierarchies and generalized topological and shape knowledge.

4.2. Abstraction of a spatial layout

A *hierarchical approach* to layout design problems has been suggested by various design researchers and used in design practice (Eastman, 1973; Pfefferkorn, 1975; Carlson & Fireman, 1987; Cort & Hills, 1987). In this approach, a layout is generated hierarchically from groups of spaces (abstract) to smaller groups (more specific) and so on until the layout is completely determined. In other words, the design process proceeds from an abstract to a more detailed layout, forming hierarchical levels of layout abstractions. With such hierarchical levels of abstractions, design can be carried out at each level of abstraction. For example, in designing a building layout, design can be carried out at the levels of floors, groups of functional spaces, individual spaces, and furniture within the spaces. Abstraction of a past spatial layout into a hierarchy therefore provides the designer with layout abstractions with which he or she can carry out his or her design at any level.

In the abstraction of a layout, learning is involved when generalizing parts of the layout that have common aspects or attributes. For example, when abstracting a building layout from the level of individual spaces to that of the groups of functional spaces, individual spaces with the same functions are generalized, forming groups of functional spaces.

Based on the importance of abstraction in design as addressed above, in the CV–S approach, the abstraction of a layout into a hierarchy based on the designer's needs is presented. That is, the abstraction is based on particular aspects or viewpoints that the designer wishes to focus on. In the spatial design process, when generating the design solution, the designer applies some design features, for example, the functions, sizes, shapes of and degree of importance of adjacencies between the spaces, and considerations, for example, the grouping of spaces based on their functions. To provide the designer with different perspectives of the abstraction, four viewpoints have been defined in the CV–S approach: *area* (the areas of spaces), *function* (the functions of spaces, e.g., *food preparation*, *store*, *shop*), *type* (the class of spaces, e.g., *private*, *public*, and *circulation*), and *closeness rating* (the degree of importance of adjacencies between spaces). Figure 7 illustrates the abstraction of a layout based on these four viewpoints. The initial layout has spaces denoted with letters from a to s.

In Figure 7, in the area viewpoint where the abstraction starts, for example, with spaces whose area are minimum, on the first level of abstraction spaces e, f, and g, and h, i, j, k, l, and m have been abstracted into two abstract spaces: (e f g) and (h i j k l m), respectively. On the next level of abstraction, spaces n and o, and p, q, r, and s have been abstracted into two more abstract spaces: (n o) and (p q r s), respectively.

In the function viewpoint, the abstraction of the layout is based on the adjacent spaces whose functions are the same. For example, spaces h, i, j, k, l, and m have the same function: 1. Therefore, on the first level of abstraction they are abstracted into an abstract space whose function is 1.

In the type viewpoint, the adjacent spaces are abstracted into classes to which the spaces belong. These classes may be termed with different names, such as private, public, and circulation areas. For example, in a house the types of the bedroom and dining room may be called, say, private areas; thus, if they are adjacent, they can be abstracted into one larger space, private area. In Figure 7, spaces 2, 3, 4, and 5, for example, have the same space type, that is, B. Consequently, when abstracting these spaces, an abstract space whose type is B is generated.

In the closeness rating viewpoint, the abstraction of the layout is based on the degree of closeness (the importance of adjacencies) between the spaces based on a particular aspect, for example, ease of movement, visual communication, aural communication. The higher the degree, the more important the adjacency between them. Therefore, the abstraction starts from the spaces with the highest degree of closeness. For example, in Figure 7, on the first level of abstraction, spaces h, i, j, k, l, and m, and p, q, r, and s have been abstracted into two abstract spaces based on the highest degree of closeness.

Like the generalization of layouts, the abstraction of a layout gives rise to several *key* issues that need to be taken into account. These key issues are outlined briefly as follows:

- *How to represent the layout?* Representations of the layout should support the abstraction processes based on four viewpoints: *area*, *function*, *type*, and *closeness rating.* For example, for the abstraction based on *area*, each space of the layout should be geometrically represented, including its coordinate points, dimensions, and area.
- *How to abstract the layout?* This involves determining the methods to be used for the abstraction processes based

https://doi.org/10.1017/S0890060498122060 Published online by Cambridge University Press

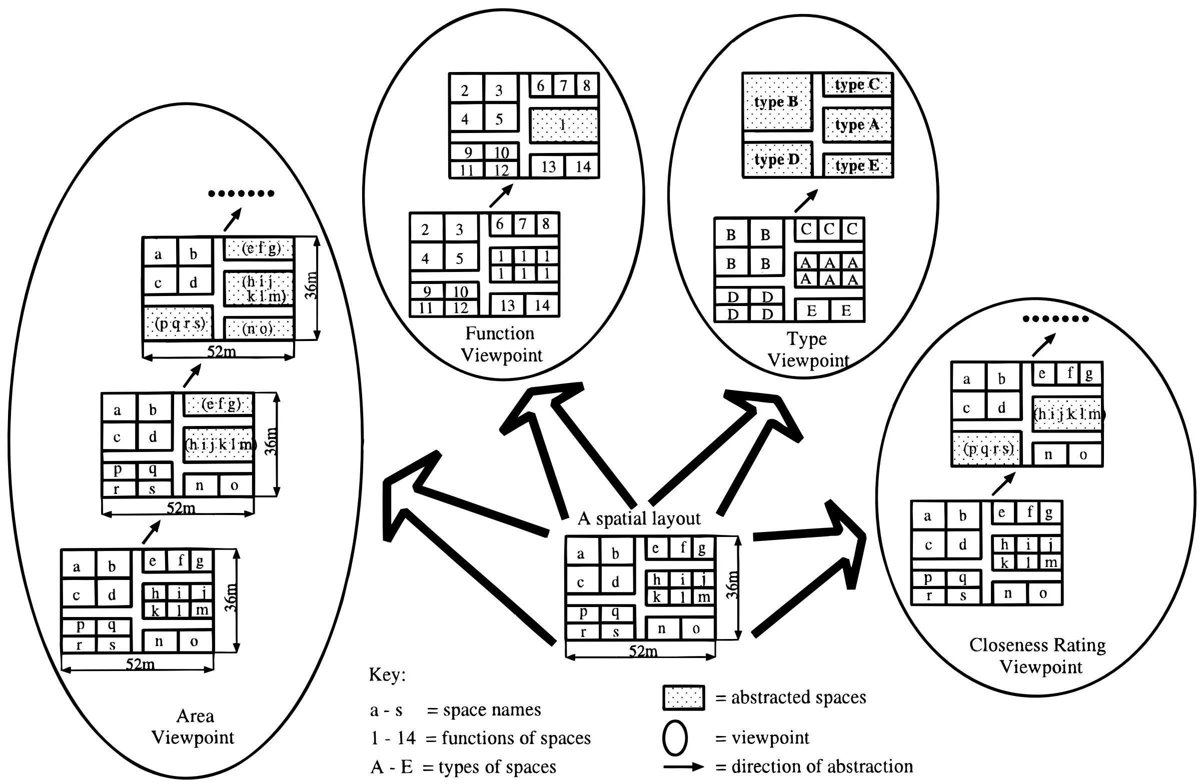


Fig. 7. An illustration of abstraction of a spatial layout in four different viewpoints.

on the four viewpoints and how the processes are to be carried out.

- *How to generate hierarchical abstractions of the layout?* It should be possible for hierarchical abstractions of the layout to be generated, so that the designer can utilize any level of abstraction that he or she wishes.

5. IMPLEMENTATION TECHNIQUES FOR THE CV–S APPROACH

In this section, techniques used to implement the CV–S approach within a computer support environment are briefly described. They are divided into two parts, following the two parts of the CV–S approach: the generalization of layouts and the abstraction of a layout.

5.1. Generalization of layouts

The techniques used to generalize layouts consist of those for pattern-matching and clustering and generalizing layouts. As for pattern-matching, the use of the *symbolic* method does not directly support the generalization of the topological patterns and the combination of the topological patterns and geometric shapes of layouts. This method currently is used to match symbolic representations of layouts, that is, design attributes, rather than layout solutions in the form of layout drawings. The use of *neural networks* has drawbacks, as addressed in Section 4.1.

Considering the above drawbacks, we adopt techniques of topological and geometric pattern-matching, that is, the "association graph technique" (Ballard & Brown, 1982) and "planar shape matching" (Leu & Huang, 1988), respectively. We recognize that there are more recent techniques of these kinds of pattern-matching (see, for example, Hanyu et al., 1992; DellaCroce & Tadei, 1994; Corno et al., 1995 for topological pattern-matching, and Niblack & Yin, 1995; Cohen & Guibas, 1997 for geometric pattern-matching). However, some of the techniques are application-specific and some others have similar properties as, but do not necessarily improve, the techniques we adopt. The techniques we adopt are more generic, more basic, and thus more applicable to different problems of these kinds of pattern-matching. In addition, in the implementation of the pattern-matching parts of the CV–S approach, these techniques work appropriately, in that there are no significant problems with time complexity of the pattern-matching process and the techniques are capable of producing good results.

The application of the adopted techniques provides an accurate matching result, since they can define the minimum measure of the difference between topological and geometric patterns. In other words, the maximum preservation of the complete information regarding matching topological or geometric patterns can be achieved. The "association graph technique" is used as part of the techniques for topological generalization. The combination of this technique and the "planar shape matching" is used as part of the techniques for combined topological and geometric generalization. These two techniques are briefly described in Sections 5.1.1 and 5.1.2.

As for clustering and generalizing layouts, we are currently investigating, developing, and attempting to implement existing techniques within a computer system to evaluate the CV–S approach.

5.1.1. Association graph technique–based topological pattern matching

The "association graph technique" employs a data structure called an "association graph" and a "clique-finding algorithm" (Ballard & Brown, 1982). The technique is able to find all of the isomorphisms (similarities) between the *subgraphs* of a graph and the *subgraphs* of another graph. Thus it allows the "best match" between two graphs to be defined, as compared to techniques for defining graph isomorphism between two graphs that aim to find an exact similarity between the graphs where nodes or arcs must not be missing from one or the other graph.

An "association graph" is a graph whose nodes are pairs of nodes of two graphs of matched layouts, which have similar properties, and edges connect the graph's nodes that represent *compatible* assignments. The "best match" between the two graphs can be defined from the "association graph." This is achieved by finding the largest set of node correspondences in the association graph that are all mutually compatible under the relations. This means finding the largest totally connected set of nodes, which is termed a *clique*. The "clique-finding algorithm" can find cliques with the largest number of nodes.

5.1.2. Planar shape matching

The "planar shape matching" technique combined with the "association graph" is used for the combined topological and geometric pattern matching. In this combined form of matching, the "association graph technique" described above initially is applied to match the topological patterns of layouts. The "planar shape matching" technique then is applied to match the shapes of any corresponding nodes (spaces) in the largest clique of nodes resulting from the topological pattern matching. The result of shape matching of each of the corresponding nodes is accumulated to give a measure of the overall shape similarity between layouts.

The "planar shape matching" technique developed by Leu and Huang (1988) is adopted, since it allows the shape representation and matching to be view-independent. That is, they do not depend on the locations from which the shape is viewed. In the shape representation, the shape is converted into a different representation, which is view-independent. The shape matching or recognition is also view-independent, since the technique allows viewing transformations between shapes to be matched, which make the shape matching or recognition invariant to shape rotation, scaling, translation, and skewing.

In the above shape matching technique, it is assumed that shapes are planar, their boundaries are closed, and they are without self-crossings. For shape matching, a shape is represented in a *binary tree* structure whose nodes represent triangles resulting from the partition of the shape. A binary tree is a tree in which every node has two other nodes as its children. A simple example is given in Figure 8. The solid lines of the shape in Figure 8a are the boundary of the shape. The shape is partitioned into triangles, aiming to simplify the shape into a quadrilateral. In the first step of the partition, three triangles (C, D, and E) are formed. Merging the sides of each of these triangles and replacing them with their base lines (the broken lines excluding line S) simplifies the shape into a quadrilateral. The next step of the shape partition is to divide the quadrilateral by line S into two triangles: A and B.

Having partitioned the shape in Figure 8a into triangles, a binary tree is constructed (see Figure 8b). The root node of the tree is the dividing line S and the children of S are triangles A and B. Triangles C and D attach to the left and right sides of triangle A, respectively. Therefore, in the tree the two triangles are the left and right children of triangle A. Similarly, triangle E attaches to the left side of triangle B, while there is no triangle on the right side of triangle B. Therefore, in the tree triangle, E is the left child of triangle B, while the right side of triangle B is empty. The "+" and "−" signs indicate whether a triangle lies inside or outside its parent triangle. In addition to these signs, the following three attributes are recorded in each node:

- The area of the triangle.
- The ratio between the height of the triangle, h, and the base of the triangle, a (i.e., h/a).
- The ratio between the projection of the left side of the triangle on the base, b, and the base (i.e., b/a).

Matching two shapes is realized by comparing the two corresponding binary tree representations. In this case, the two trees are compared in a top-down fashion, that is, they are compared node by node from the root nodes through to the internal nodes to the leaf nodes. The process of comparing the two trees is carried out according to the *breadth-first*

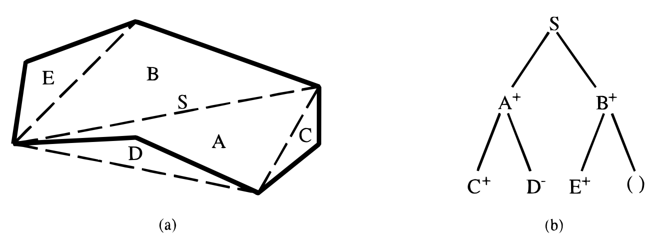


Fig. 8. Shape partition: (a) a shape partitioned into triangles; (b) the resulting binary tree of shape partition in (a) (Leu & Huang, 1988).

https://doi.org/10.1017/S0890060498122060 Published online by Cambridge University Press

search sequence (Leu & Huang, 1988). In each node comparison the node (shape) difference with regard to the four attributes of each node is recorded and added to the cumulative node difference obtained so far. Thus the result of matching the two trees (shapes) is the final cumulative shape difference, which indicates the value of dissimilarity between the two shapes. This value is numerical, but it can be transformed into a qualitative measure representing a certain degree of similarity. Classes of the degree of similarity between objects as defined by Smith (1989) (see Section 3.2) can be used for this purpose.

Having matched the shapes of all of the corresponding spaces, the resulting dissimilarity measures are summed up to give a measure of the overall shape dissimilarity between layouts.

5.2. Abstraction of a layout

The abstraction of a layout is realized by merging some or all of the adjacent spaces into merged spaces. In the CV–S approach, spaces are represented in a coordinate system and only rectangular spaces, and polygonal spaces that can be represented as a collection of rectangles, are of concern to us. Consequently, the abstraction based on the above four viewpoints also represents the geometrical abstraction of a layout. For merging spaces and measuring the area of the merged space, we adopt methods of the "geometry of rectangles" (Preparata & Shamos, 1985), namely, the *contour* and *measure of a union of rectangles.* We recognize that there are a few number of more recent techniques for measuring the area of and defining the contour of the union of rectangles (see, for example, Widmayer & Wood, 1987; Wu et al., 1988; Datta, 1997). However, they have similar properties as the methods we adopted. In the implementation of the abstraction part of the CV–S approach, the methods we adopt work well. That is, they provide the accurate result of merging spaces in that the contour and area of the merged space can be accurately produced. The use of these methods is combined with the use of a data structure called the *segment tree* (Preparata & Shamos, 1985), a rooted binary tree that represents intervals of integers. This tree is used to represent the ordinates of the vertical sides of rectangles that represent spaces to be merged.

6. SPIDA SYSTEM

To evaluate the CV–S approach, a computer system called SPIDA has been developed. This system is implemented using Harlequin LispWorks (Common Lisp and CLOS) (The Harlequin Group Limited, 1994) running on a Silicon Graphics or Sun Sparc workstation. For the processes of generalizing spatial layouts, tools for matching topological patterns of layouts and the combined topological patterns and geometric shapes of layouts have been developed. Tools for clustering and generalizing layouts currently are under development. Tools for the processes of abstracting a spatial layout based on the *area*, *function*, *type*, and *closeness rating* also have been developed. In this section, we present some experimental results of evaluating the tools that have been developed.

6.1. Topological pattern-matching of layouts

The pattern-matching operation involved in the topological generalization of layout design cases is topological pattern-matching between the cases, aiming to define the similarity in the topological pattern between one design case and another. Thus, the operation does not involve matching the cases against an input layout. However, for the purpose of evaluating the "association graph technique" adopted for the topological pattern-matching, in this section we present and briefly discuss some experimental results on matching design cases against input layouts. For the experiment, a set of seven ship layout cases, based on existing designs (see Figure 9), have been stored in the system. A set of three ship layout templates (see Figure 10) also has been set as the input to the pattern-matching process. The term "template" is commonly used in the field of pattern recognition to mean an input pattern to the pattern-matching process (Hall & Matias, 1993).

Each space of layout design cases and templates in Figures 9 and 10, respectively, is labeled with the nature of the space. The results of topological pattern-matching between each of the templates and the layout cases are presented in Table 1. In this table, for each template, the layout cases are ordered from the most (on the top) to the least (at the bottom) similar to the template. This order of layout cases is based on the resulting number of corresponding spaces. In this case, the higher the number of the corresponding spaces, the more similar is the layout case to the template. If there is more than one layout case that has the same number of the corresponding spaces, they have the same rank. For example, in Table 1, for ship layout template 1, ships 3 and 6 are placed in the same box and on the top of the order of ship layout cases, since they have the same, highest number (7) of corresponding spaces. The nature of the space of each of the corresponding spaces is also included in the table.

From the above experimental results (see Table 1) it can be observed that the number of corresponding spaces between a layout case and template represents the degree of similarity between them. The degree of similarity may be classified into classes defined by the designer, where each class may have more than one different number of corresponding spaces. Classes of degree of similarity as defined by Smith (1989) (see Section 3.2), for example, may be used for this purpose, by assigning a particular class, for example, the overall similarity, part similarity, and so on, to one or more than one different number of corresponding spaces. Alternatively, each number of corresponding spaces may represent a class of degree of similarity.

For the topological generalization, layout design cases are clustered based on classes of degree of similarity between them. For the purpose of showing how the clustering of the

https://doi.org/10.1017/S0890060498122060 Published online by Cambridge University Press

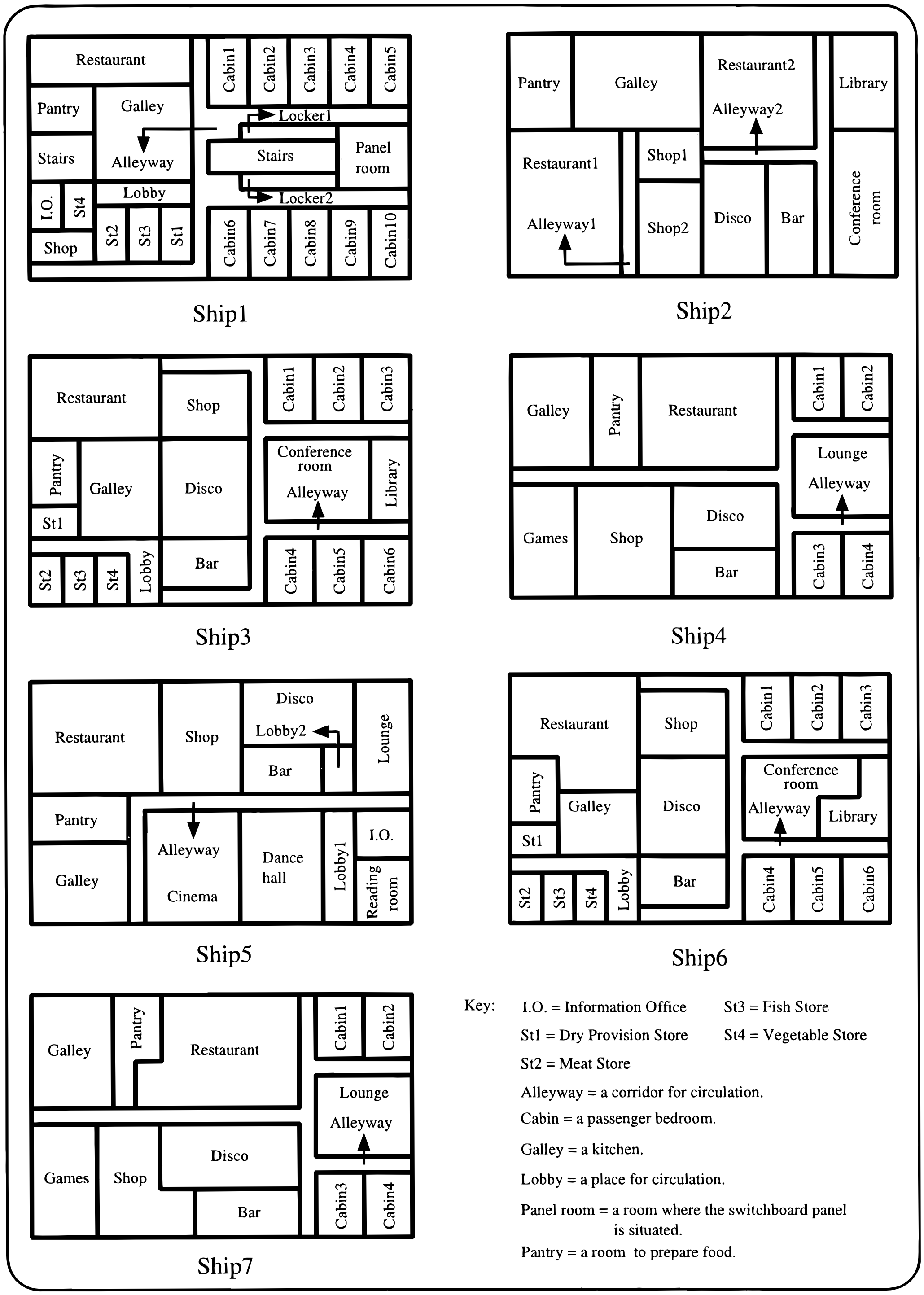


Fig. 9. A set of seven ship layout cases.

cases can be carried out, the clustering of the cases based on the experimental results given in Table 1 may be considered. The clustering is based not only on the similarity in the number of corresponding spaces between each of the layout cases and a template, but also on the similarity in their corresponding spaces. Thus, in relation to the template, a case with a certain number of corresponding spaces that is the same as that of another case, but with different corresponding spaces, cannot be clustered with the latter case. For example, in the results of matching layout cases against ship layout template 2, ships 3 and 6 can be grouped into a cluster since they have the same corresponding spaces in

https://doi.org/10.1017/S0890060498122060 Published online by Cambridge University Press

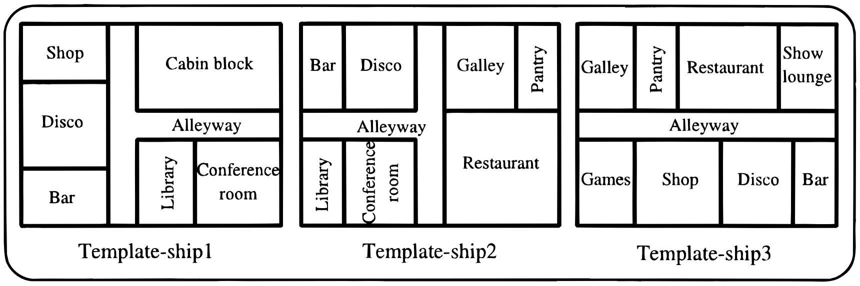


Fig. 10. A set of three ship layout templates.

relation to the template. On the other hand, although ship 2 has the same number of corresponding spaces as that of the cluster, this ship is not grouped into the cluster, since it has different corresponding spaces from those of the cluster.

Clustering the cases can lead to the generalization of the cases. That is, a cluster is created based on what are common between the cases as its elements. A concept that represents the commonalities between the cases then can be generated. This concept consequently represents the generalization of the cases.

6.2. Combined topological pattern and geometric shape matching of layouts

For the combined topological and geometric generalization of layout design cases, the pattern-matching operation involved is the combination of topological pattern and geometric shape matching between the cases. However, as with the topological pattern-matching discussed in Section 6.1, for the purpose of evaluating the combination of the "association graph technique" and the "planar shape matching"

Table 1. *Results of topological pattern-matching between each of the layout templates in Figure 10 and the layout cases in Figure 9*

Layout templates	Ordered ships	Number of corresponding spaces	Corresponding spaces
Ship layout template 1	Ship 3 Ship 6	7	(Shop, Disco, Bar, Library, Conference-room, Cabin, Alleyway)
	Ship 2	6	(Shop, Disco, Bar, Library, Conference-room, Alleyway) (Note: Shop1 of Ship2)
	Ship 4 Ship 7	4	(Disco, Bar, Cabin, Alleyway)
	Ship 1	3	(Shop, Cabin, Alleyway)
	Ship 5	2	(Bar, Alleyway)
Ship layout template 2	Ship 2	7	(Disco, Bar, Library, Conference-room, Restaurant, Galley, Pantry) (Note: Rest.1 of Ship2)
	Ship 3 Ship 6		(Disco, Bar, Library, Conference-room, Restaurant, Pantry, Alleyway)
	Ship 1 Ship 4 Ship 5 Ship 7	4	(Restaurant, Galley, Pantry, Alleyway) (Disco, Bar, Galley, Alleyway) (Disco, Bar, Restaurant, Pantry) (Disco, Bar, Galley, Alleyway)
Ship layout template 3	Ship 4 Ship 7	7	(Pantry, Galley, Games, Disco, Bar, Lounge, Alleyway)
	Ship 2 Ship 5	5	(Pantry, Galley, Shop, Disco, Bar) (Pantry, Galley, Bar, Lounge, Alleyway)
	Ship 3 Ship 6	4	(Shop, Disco, Bar, Alleyway)
	Ship 1	3	(Galley, Shop, Alleyway)

technique for this operation, in this section we present and briefly discuss some experimental results on matching layout design cases against layout templates. For the experiment, the layout cases and templates used in topological pattern-matching (see Figures 9 and 10) are used. The results of matching each of the layout templates against layout cases are presented in Table 2. In this table, the layout cases first are ordered based on the number of resulting corresponding spaces. They then are ordered based on shape dissimilarity measures. The higher the measure, the more dissimilar (less similar) the layout case to the template. The resulting corresponding spaces are the same as those in topological pattern-matching (see Table 1).

As with topological pattern-matching, in this kind of pattern-matching, layout cases can be clustered using classes of degree of similarity. One or more than one different number of corresponding spaces can be used as a class of degree of similarity. Classes of degree of similarity as defined by Smith (1989) may be used for clustering the cases. The difference between this kind of pattern-matching and the topological pattern-matching is that the clustering is not only based on the same corresponding spaces, but also on the overall shape dissimilarity measures between the spaces. In this case, the clustering may be based on particular ranges of these measures. For the purpose of showing how the clustering of the cases can be carried out, the clustering of the cases based on the experimental results given in Table 2 may be considered. For example, in the results of matching layout cases against ship layout template 2, ships 3 and 6, which have the same corresponding spaces, may be clustered together because the difference between their shape dissimilarity measures is within the range, for example, 75%. Using clusters of the cases resulting from this kind of pattern-matching, generalization of the cases based on their combined topological patterns and geometric shapes can be carried out.

Table 2. *Results of combined topological pattern and geometric shape matching between each of the layout templates in Figure 10 and the layout cases in Figure 9*

Layout templates	Ordered ships	Number of corresponding spaces	Shape dissimilarity measures
Ship layout template 1	Ship 3	7	4.960
	Ship 6	7	6.317
	Ship 2	6	3.784
	Ship 4	4	3.376
	Ship 7	4	3.482
	Ship 1	3	4.587
	Ship 5	2	2.473
Ship layout template 2	Ship 2	7	2.345
	Ship 3	7	4.774
	Ship 6	7	6.328
	Ship 5	4	0.747
	Ship 4	4	2.976
	Ship 7	4	3.336
	Ship 1	4	5.589
Ship layout template 3	Ship 4	7	2.798
	Ship 7	7	4.453
	Ship 2	5	1.311
	Ship 5	5	2.084
	Ship 3	4	3.868
	Ship 6	4	3.868
	Ship 1	3	4.724

6.3. Abstraction of a spatial layout

As was mentioned in Section 4.2, the abstraction of a spatial layout is based on four viewpoints: *area*, *function*, *type*, and *closeness rating*. The implementation of each of the abstractions in the SPIDA system is presented in this section. For all of these abstractions the layout of Ship 5 in Figure 9 is used for the example.

6.3.1. *Abstraction based on area*

The abstraction of a layout based on *area* uses the area of each space of the layout as the aspect for the space merging process. In the abstraction process, the system initially finds a space with the minimum area. It then finds any other space whose area is within an area range with respect to the found minimum area. The area range is input by the user. Next, the system merges each of the spaces whose areas are within the area range with its adjacent spaces whose weights (area differences with this space) are minimum. The process continues, resulting in the different levels of abstraction until a layout of a single merged space is achieved. Alternatively, the user can input any number of levels of abstraction according to his or her needs.

As an example, Figure 11 illustrates the abstraction of the layout of Ship 5 based on *area*. The original layout of Ship 5 is at the bottom of the figure. The area range that the user input for this abstraction is 100%. The abstraction process results in four levels of abstractions. It can be seen that on the first level of abstraction spaces 6 and 7 are merged into a merged space labeled MS-177-1, and spaces 9, 10, and 11 are merged into MS-178-1. The label of a merged space is automatically generated by the system, and the last number of the label shows the level of abstraction. At the top level of abstraction (level 4), a single space (MS-183-4) is achieved and displayed at the top of the figure.

6.3.2. *Abstraction based on function*

In the abstraction of a layout based on *function*, the layout is abstracted by merging the adjacent spaces whose functions are the same. Figure 12 illustrates the abstraction of the layout of Ship 5 based on the functions of the layout spaces. The original layout is at the bottom of the figure. In this layout abstraction, spaces MS-129-1 and MS-131-1, for exam-

https://doi.org/10.1017/S0890060498122060 Published online by Cambridge University Press

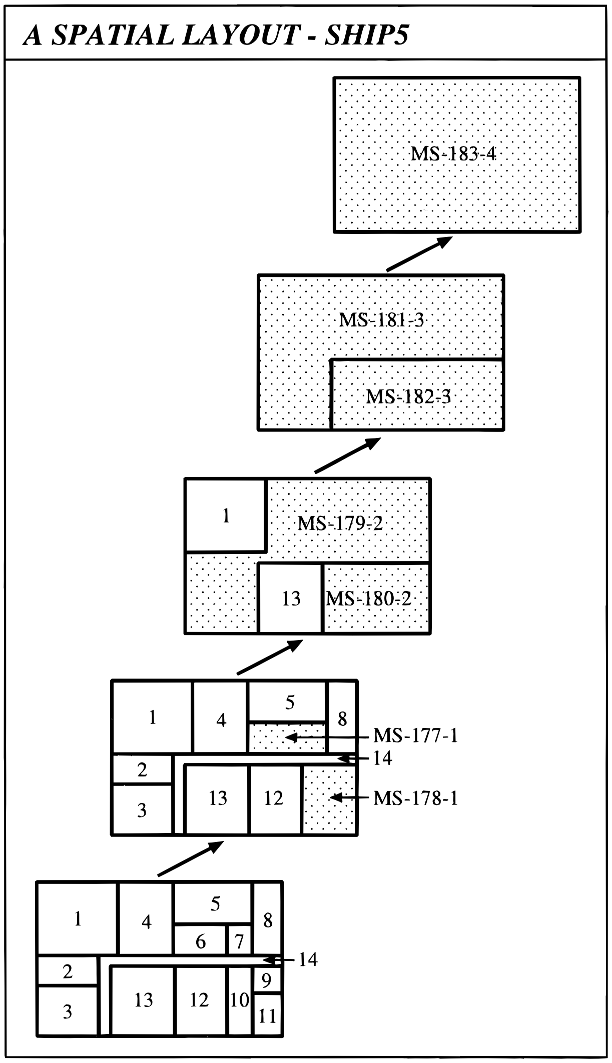


Fig. 11. Abstraction of the layout of Ship 5 based on area.

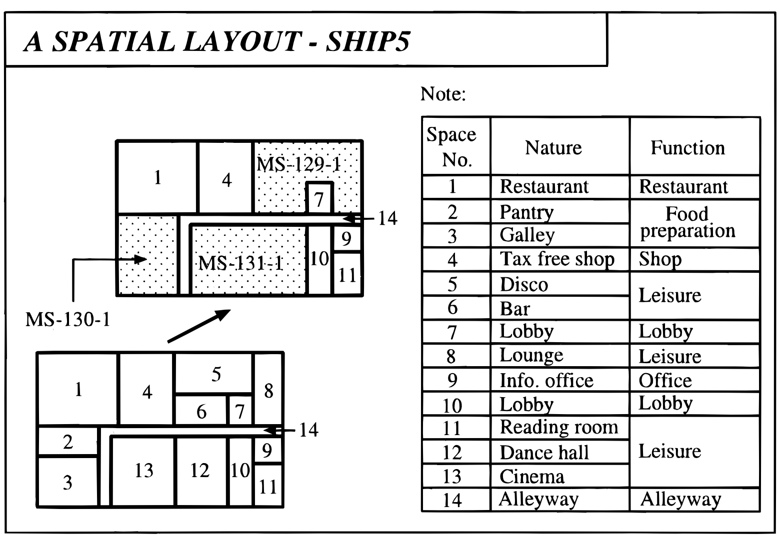
ple, are the results of merging spaces 5 (the disco), 6 (the bar), 8 (the lounge), 12 (the dance hall), and 13 (the cinema), respectively, since they have the same function: *leisure*. Similarly, spaces 2 (the pantry) and 3 (the galley) are merged into a merged space, MS-130-1, since their functions are the same: *food preparation*. Furthermore, a number of levels of abstraction also may be generated by abstracting the layout partially. In this case, the user can control the abstraction by inputting parts of the layout (i.e., particular spaces) whose functions are the same. This process can be repeated with the other parts until all of the spaces with the same functions have been merged.

6.3.3. *Abstraction based on space type*

In the abstraction of a layout based on *type*, the layout is abstracted by merging the adjacent spaces whose types or groups are the same. Figure 13 illustrates the abstraction of the layout of Ship 5 based on the types or groups of spaces. The original layout is at the bottom of the figure. In this layout abstraction, spaces 1 (the restaurant), 4 (the tax free shop), 5 (the disco), 6 (the bar), and 8 (the lounge), for example, have been merged into a merged space, MS-136-1, since they can be grouped as the *public area*. Consequently, the abstraction based on *type* can be applied to the abstract layout resulting from the abstraction process based on the functions of the spaces. In addition, as in the abstraction of a layout based on the functions of spaces, the abstraction based on space type can be carried out partially and repeatedly, resulting in abstractions at different levels.

6.3.4. *Abstraction based on closeness rating*

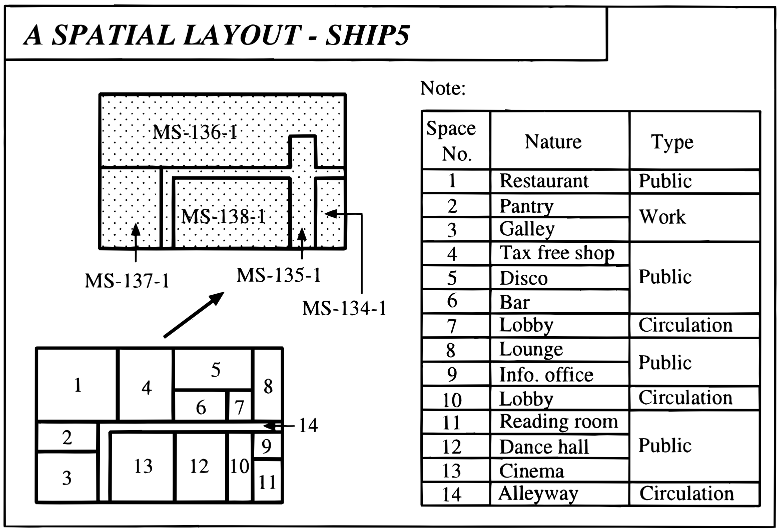
In the abstraction of a layout based on *closeness rating*, the layout is abstracted in different levels of abstractions by merging the adjacent spaces starting from the highest close-



Space No.	Nature	Function
1	Restaurant	Restaurant
2	Pantry	Food preparation
3	Galley	
4	Tax free shop	Shop
5	Disco	Leisure
6	Bar	
7	Lobby	Lobby
8	Lounge	Leisure
9	Info. office	Office
10	Lobby	Lobby
11	Reading room	Leisure
12	Dance hall	
13	Cinema	
14	Alleyway	Alleyway

Fig. 12. Abstraction of the layout of Ship 5 based on space function.

https://doi.org/10.1017/S0890060498122060 Published online by Cambridge University Press



Space No.	Nature	Type
1	Restaurant	Public
2	Pantry	Work
3	Galley	
4	Tax free shop	Public
5	Disco	
6	Bar	
7	Lobby	Circulation
8	Lounge	Public
9	Info. office	
10	Lobby	Circulation
11	Reading room	Public
12	Dance hall	
13	Cinema	
14	Alleyway	Circulation

Fig. 13. Abstraction of the layout of Ship 5 based on space type.

ness ratings between them. For this abstraction, seven grades of closeness ratings are defined. Of these seven grades, six grades are represented in numbers ranging from 0 to 5, with 0 being *not adjacent* and 5 being *essential*. Thus, the higher the grade, the more important the adjacency between spaces. Another grade is denoted with the letter E (*exception*). This grade is meant here to be the grade of the adjacency between a space for a particular activity, for example, the galley, pantry, and so on, and a space for circulation, for example, the alleyway. In this abstraction, it is assumed that adjacent spaces with the grade E will not be merged, since they are used for different purposes, that is, activities and circulation. However, adjacent spaces whose functions are for circulation will be merged.

Figure 14 illustrates the abstraction of the layout of Ship 5 based on *closeness rating*. The original layout is at the bottom of the figure. The abstraction process results in five levels of layout abstractions. In the first level of abstraction, for example, spaces 7, 10 (the lobbies), and 14 (the alleyway), and 1 (the restaurant), 2 (the pantry), and 3 (the galley), are merged into two merged spaces, MS-144-1 and MS-145-1, respectively, since the grade of the closeness rating between each of these spaces and another is the highest (5), compared to the other closeness ratings.

7. DIMENSIONS OF LEARNING

In this section, we evaluate the work we presented in the previous sections in the light of dimensions of learning in design defined by Grecu and Brown (1996).

7.1. What can trigger learning?

The motivation behind the development of the SPIDA system is to improve the ability of layout design learning systems. Existing CBLL systems generalize layout design cases into single levels of abstractions. They are not able to generalize design cases into a hierarchy. In addition, they do not abstract a layout into a hierarchy. The SPIDA system is able to carry out such generalization and abstraction, thus improving the abilities of the existing CBLL systems. With such improvements, the abstract and generalized knowledge of layout designs can be provided in the different levels for a new design session.

In the CV–S approach, learning is triggered by the designer's need to learn about the domain, domain exploration (Duffy et al., 1995), or acquire knowledge for a new design problem. The approach presented in this paper supports a number of viewpoints for learning, controlled and triggered by the designer.

7.2. What are the elements supporting learning?

Since in the SPIDA system layouts are abstracted and generalized into different levels, there are two forms of layout design knowledge that are elements which support learning. The first form is layout design cases stored in the system, and the second form is abstract and generalized knowledge in different levels, resulting from dynamically abstracting and generalizing the knowledge.

7.3. What might be learned?

In the generalization of layouts, structural descriptions of layouts in the form of topological patterns and combined topological patterns and geometric shapes are learned. In the abstraction of a layout where learning is involved, when generalizing parts of the layout based on the above four viewpoints, numerical and symbolic attributes of these parts are learned. New abstract concepts are created which reflect four

https://doi.org/10.1017/S0890060498122060 Published online by Cambridge University Press

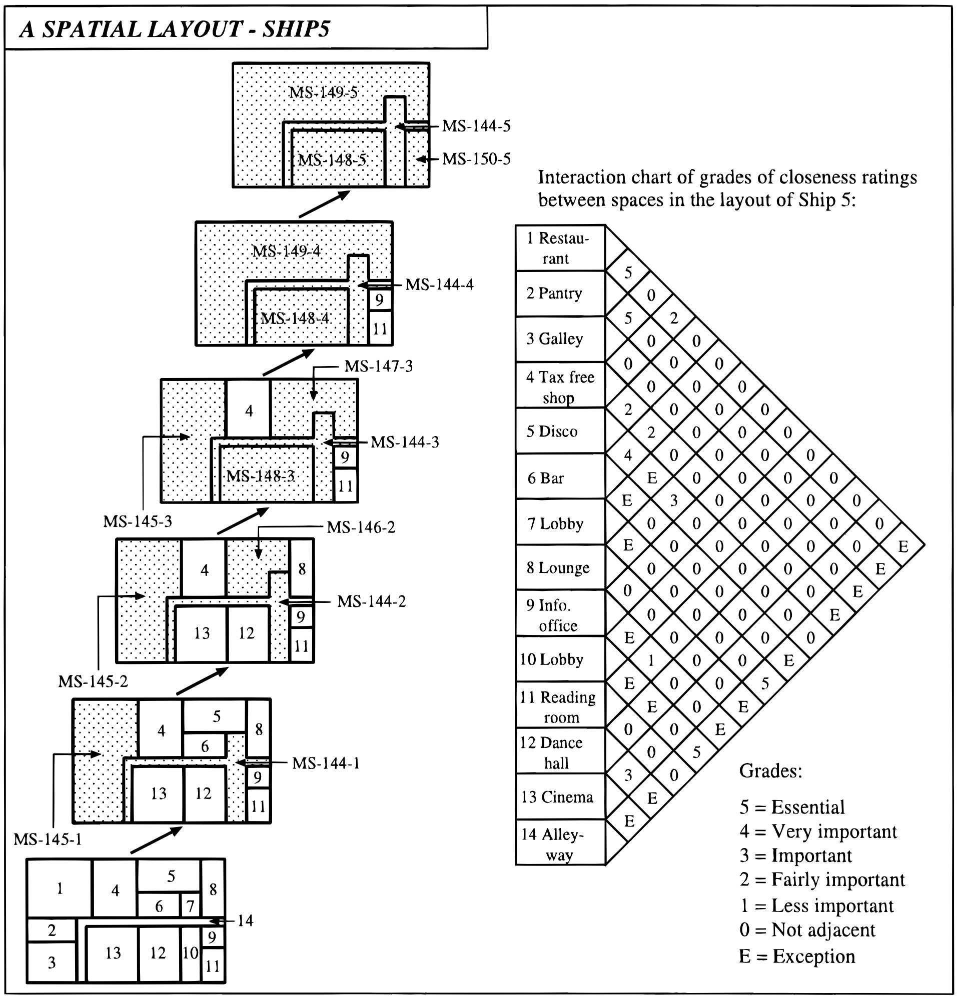


Fig. 14. Abstraction of the layout of Ship 5 based on closeness rating.

particular aspects of spatial knowledge: the area, space function, space type, and closeness rating.

7.4. Availability of knowledge for learning

Repositories of layout design cases are available within the SPIDA system as specific design knowledge for learning. They are dynamically updated to also store abstract and generalized knowledge resulting from the learning process. The knowledge is in different levels of abstraction, following the learning process, which is carried out incrementally. Thus, knowledge in one level is available for learning to produce more abstract knowledge on the level above it.

7.5. Methods of learning

Methods of learning used in the SPIDA systems are induction, abstraction, and acquisition. *Induction* is used when the system infers the general description from layout design instances, that is, generalizing layouts. *Abstraction* is used to abstract specific layout design knowledge into hierarchical levels of layout abstract knowledge. *Acquisition* is used to acquire new knowledge generated from the processes of abstraction and generalization.

7.6. Local versus global learning

Since learning processes occur in one system, that is, the SPIDA system, learning is done locally. This means that the learned knowledge, that is, abstract and generalized layout knowledge, is used locally within the system for subsequent learning to generate a hierarchy of layout knowledge.

7.7. Consequences of learning

SPIDA generates new knowledge that can be used to improve new design problem solving. In particular, the new knowledge that can be acquired by SPIDA is abstract and generalized knowledge of topological patterns and of the combined topological patterns and geometric shapes of layouts, and layout abstract knowledge resulting from the abstraction of a layout based on the above four viewpoints. With the new knowledge, the quality of designs can be improved, since the knowledge is more widely applicable for a new design session. That is, using the new knowledge, the designer can explore a potentially larger number of design solutions.

8. CONCLUSION

In this paper, we have presented an approach to abstracting and generalizing layout design cases. We have shown that this approach (customized viewpoint—spatial) overcomes the limitations that existing CBLL systems have. That is, it supports the generalization of layouts and the abstraction of a layout into hierarchical levels of abstractions. Furthermore, in the generalization of layouts, techniques for topological pattern and geometric shape matching have been presented. Using these techniques, this part of the approach supports graphic information retrieval of past spatial layout design cases and their abstractions, which is not supported by existing CBLL systems. Provided with such information, the designer will have topological patterns and geometric shapes of specific and abstract spatial layouts that can be used as starting solutions for a new design problem.

We also have presented the SPIDA system to evaluate the CV–S approach. What we have demonstrated is the basic parts of the generalization of spatial layouts and the completion of the abstraction of a layout. As the starting steps of the generalization, we have finished developing the tools for topological pattern-matching and the combined topological pattern and geometric shape matching of layouts. The next steps are to develop the tools for clustering and generalizing layouts based on both topology and the combined topology and geometry.

As for the abstraction of a layout, the SPIDA system allows the user to select a particular layout according to the user's needs. We have defined four different viewpoints: *area*, *function*, *type*, and *closeness rating*, on which the abstraction may be based. The number of levels resulting from the abstraction process of a layout may be generated automatically by the system or set by the user. The system can save all or a number of the levels of the layout abstraction, according to the user's needs. They can be retrieved and subsequently used for a new design problem.

Finally, with its state of development, we believe that the SPIDA system supports learning from design cases, since to generalize the design cases we need to pattern-match them in order to define their similarities. The system also supports learning a layout design case, in that it generalizes parts of the layout based on particular viewpoints and generates a hierarchy of layout abstractions. Further work, that is, clustering and generalizing layouts, however is still required to fully implement the CV–S approach in the SPIDA system.

REFERENCES

Akin, O. (1978). How do architects design? In *Artificial Intelligence and Pattern Recognition in Computer Aided Design* (Latombe, J.L., Ed.), pp. 65–103. North-Holland, The Netherlands.

Ballard, D.H., & Brown, C.M. (1982). *Computer Vision.* Prentice-Hall, Inc., Englewood Cliffs, NJ.

Bareiss, R., Ed. (1991). *Proceedings: Case-Based Reasoning Workshop 1991.* Morgan Kaufmann, Los Altos, CA.

Carlson, C.M., & Fireman, H. (1987). General arrangement design computer system and methodology. *Naval Engineers Journal 99(3)*, 261–273, May.

Cohen, P.R., & Feigenbaum, E.A. (1982). *The Handbook of Artificial Intelligence*, Vol. 3. Pitman Books Limited, London.

Cohen, S.D., & Guibas, L.J. (1997). Partial matching of planar polylines under similarity transformations. *Proc. Ann. ACM-SIAM Symp. on Discrete Algorithms*, Jan. 5–7, New Orleans, 777–786.

Corno, F., Prinetto, P., & SonzaReorda, M. (1995). Using symbolic techniques to find the maximum clique in very large sparse graphs. *Proc. European Design and Test Conf.*, March 6–9, Paris, 320–324.

Cort, A., & Hills, W. (1987). Space layout design using computer assisted methods. *Naval Eng. J. 99(3)*, 249–260.

Coyne, R.F., & Flemming, U. (1990). Planning in design synthesis: Abstraction-Based LOOS (ABLOOS). *Proc. Fifth Int. Conf. AI in Engrg.* (Gero, J., Ed.), 1:Design.

Coyne, R.D., Newton, S., & Sudweeks, F. (1989). Modelling the emergence of schemas in design reasoning. In *Reprints of Modelling Creativity and Knowledge Based Creative Design*, pp. 173–205. Design Computing Unit, Department of Architectural and Design Science, University of Sydney.

Coyne, R.D., & Postmus, A.G. (1990). Spatial applications of neural networks in computer-aided design. *Artif. Intell. Engrg. 5(1)*, 9–22.

Darden, L. (1987). Viewing the history of science as compiled hindsight. *AI Magazine* (Summer), 33–41.

Datta, A. (1997). Geometric data structures on a reconfigurable mesh, with applications. *Proc. International Parallel Processing Symposium (IPPS 1997)*, April 1–5, Geneva, 708–712.

DellaCroce, F., & Tadei, R. (1994). Multi-KP modeling for the maximum-clique problem. *Europ. J. Operational Res. 73(3)*, 555–561.

Duffy, A.H.B. (1993). Machine learning in design (Guest editorial). *Artif. Intell. Engrg. 8(3)*, 157–158.

Duffy, A.H.B., & Kerr, S.M. (1993). Customised perspectives of past designs from automated group rationalisation. *Artif. Intell. Engrg. 8(3)*, 183–200.

Duffy, S.M., Duffy, A.H.B., & MacCallum, K.J. (1995). A design reuse process model. *Proc. Int. Conf. on Engineering Design (ICED'95)*, Aug. 22–24, Praha, 490–495.

Eastman, C.M. (1973). Automated space planning. *Artif. Intell. 4*, 41–64.

Foz, A. (1973). Observations on design behaviour in the parti. *Proc. Design Activity Conf.*

Fu, K.S., Ed. (1976). Digital pattern recognition. In *Communication and Cybernetics.* Springer-Verlag, New York.

Grecu, D.L., & Brown, D.C. (1996). Dimensions of machine learning in design at *http://cs.wpi.edu/~dcb/AID/taxonomy.html.* Adapted from Grecu, D.L., & Brown, D.C., Dimensions of learning in agent-based design, AID'96, June 1996, Stanford, CA.

Hall, G., & Matias, A. (1993). Rotation, scale and translation invariant template matching on a transputer network. *Microproc. Microsyst. 17(6)*, 333–340.

Hanyu, T., Kodama, T., & Higuchi, T. (1992). Digit-pipelined on-chip clique-finding VLSI processor for real-time 3-D object recognition. *Electron. Lett. 28(8)*, 722–724.

The Harlequin Group Limited. (1994). *LispWorks User's Guide*. Cambridge, UK.

Hoover, S.P., Rinderle, J.R., & Finger, S. (1991). Models and abstractions in design. *Design Studies 12(4)*, 237–245.

Jones, J.C. (1963). A method of systematic design. *Proc. Conf. on Design Methods* (Jones, J.C., & Thornley, D.G., Eds.). Pergamon Press, New York.

Jones, J.C. (1980). *Design Methods—Seeds of Human Futures.* John Wiley & Sons, New York.

Kerr, S.M. (1993). *Customised Viewpoint Support for Utilising Experiential Knowledge in Design.* Ph.D. Thesis, CAD Centre, Department of Design, Manufacturing and Engineering Management, University of Strathclyde, Glasgow.

Kolodner, J.L. (1993). *Case-Based Reasoning.* Morgan Kaufmann, New York.

Leu, J.G., & Huang, I.N. (1988). Planar shape matching based on binary tree shape representation. *Pattern Recog. 21(6)*, 607–622.

Manfaat, D., Duffy, A.H.B., & Lee, B.S. (1996). Review of pattern matching approaches. *Knowledge Engrg. Rev. 11(2)*, 161–189.

Maher, M.L., & Zhao, F. (1987). Using experience to plan the synthesis of new designs. In *Expert Systems in Computer-Aided Design* (Gero, J.S., Ed.), pp. 349–373. Elsevier Science Publishers B.V. (North Holland), The Netherlands.

Niblack, W., & Yin, J. (1995). Pseudo-distance measure for 2D shapes based on turning angle. *Proc. 1995 IEEE Int. Conf. on Image Processing*, Vol. 3, 352–355.

Oxman, R. (1990). Prior knowledge in design. A dynamic knowledge-based model of design and creativity. *Design Studies 11(1)*, 17–28.

Pahl, G., & Beitz, W. (1988). *Engineering Design: A Systematic Approach.* The Design Council, London, UK.

Pfefferkorn, C.E. (1975). The Design Problem Solver: A system for designing equipment or furniture layouts. In *Spatial Synthesis in Computer-Aided Building Design*, Ch. 5, pp. 98–146. John Wiley & Sons, New York.

Preparata, F.P., & Shamos, M.I. (1985). *Computational Geometry, An Introduction* (Texts and Monographs in Computer Science). Springer-Verlag, New York.

Simon, H.A. (1981). *The Sciences of the Artificial*, Ch. 7. MIT Press, Cambridge, MA.

Smith, L.B. (1989). From global similarities to kinds of similarities: The construction of dimensions in development. In *Similarity and Analogical Reasoning* (Vosniadou, S., & Ortony, A., Eds.), Ch. 5, pp. 146–178. Cambridge University Press, New York.

Voß, A., Coulon, C.-H., Grather, W., Linowski, B., Schaaf, J., Bartsch-Sporl, B., Borner, K., Tammer, E.C., Durschke, H., & Knauff, M. (1994). Retrieval of similar layouts—about a very hybrid approach in FABEL. *AID'94, Third Int. Conf. on AI in Design* (Gero, J.S., & Sudweeks, F., Eds.), pp. 625–640. Kluwer Academic Publishers, The Netherlands.

Widmayer, P., & Wood, D. (1987). Time- and space-optimal contour computation for a set of rectangles. *Inform. Processing Lett. 24(5)*, 335–338.

Wu, A.Y., Rosenfeld, A., & Bhasbar, S.K. (1988). Parallel computation of geometric properties from the medial axis transform. *Comput. Vision, Graphics, and Image Processing 41(3)*, 323–332.

Djauhar Manfaat obtained his first degree (Ir.) in Naval Architecture from the Department of Naval Architecture and Shipbuilding Engineering, Institut Teknologi Sepuluh Nopember (ITS) Surabaya, Indonesia, in 1986. He has been a lecturer in this department since 1987. He did several research projects in ship design from 1987 to 1989. In 1991, he obtained an M.Sc. in Ship Production Technology from the University of Strathclyde. He is currently taking a Ph.D. program in spatial design at the same university. His research interests are design re-use, machine learning in design, case-based reasoning in design, and pattern matching.

Alex Duffy is presently a Senior Lecturer and Director of the CAD Centre, University of Strathclyde. His main research interests are knowledge-based conceptual design, product and design knowledge modelling, machine learning and past design utilization, performance measurement and design productivity, and design coordination. He is the leader of a European-wide group working in design coordination. He is on the advisory board of AI in Design, Engineering Design, and Concurrent Engineering conferences and on the editorial board of *Research in Engineering Design and Design Computing.* He organized and chaired the Engineering Design Debate on design productivity (EDD'96) and is a member of IFIP Technical Committee Working Group 5.8 in product structuring and documentation.

Byung Suk Lee, a Senior Lecturer at the University of Strathclyde, graduated from Seoul National University with a BEng in Naval Architecture in 1970. After a brief period with the Korea Institute of Science and Technology as a research assistant, he obtained his M.Sc. in Ship Production Technology and Ph.D. in Marine Technology from the University of Strathclyde in 1976 and 1982, respectively. He then joined the academic staff of the Department of Ship and Marine Technology of the same university in 1983.