Ensuring Usability of CAAD Systems
A Hybrid Approach

Sheng-Fen Chien
National Taiwan University of Science and Technology

Key words: Usability evaluation, GOMS analysis, Usability engineering, Object-oriented software engineering

Abstract: Many CAAD software prototypes today are developed with the aim to bring research results closer to practice. This paper describes a hybrid approach that integrates an Object-Oriented Software Engineering (OOSE) methodology with a usability analysis methodology—GOMS. This approach is examined through two case studies and has shown promising results. It enables CAAD system developers to be aware of usability issues and conduct usability evaluation as early as the analysis phase of the software development process. Consequently, this may improve the quality of CAAD software systems as well as ensure the usability of the systems.

1. INTRODUCTION

The advances of CAAD research have discovered many varieties of functionality a computer can provide to assist designers. To demonstrate the functionality and its usefulness, researchers develop working software prototypes and subject them to user evaluations. According to Nielsen (1993), "usefulness" deals with the question of whether a system can be used to accomplish desired goals, and it involves two issues—utility and usability: utility focuses on the functionality of a system, and usability examines how well users can use that functionality. For a simple prototype, its user interface may be adjusted quickly and as many times as necessary to achieve its intended functionality. For a prototype with complex functionality, however, its user interface design has to focus on ensuring usability so that its functionality can be evaluated systematically.

In this paper, I present a software and usability engineering process that takes a hybrid approach to integrate the Object-Oriented Software Engineering (OOSE) methodology with the usability analysis methodology—GOMS. The objective of this hybrid approach is to enhance the evaluation phase of the software engineering process so that it covers the full range of the software lifecycle and ensures the usability of the CAAD software systems.

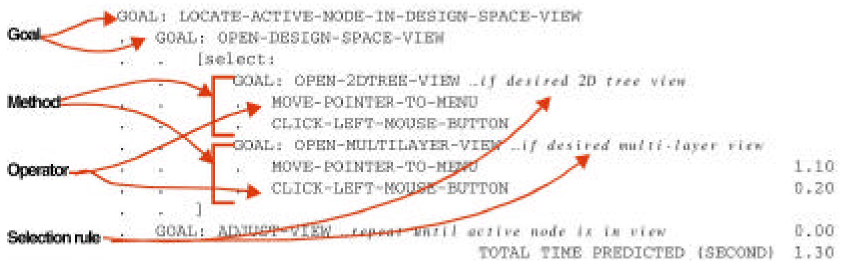
2. USABILITY AND SOFTWARE ENGINEERING

2.1 Usability Engineering: GOMS Analysis

Methods to evaluate usability can be categorized into two types—*usability analysis* and *usability testing*. Usability analysis evaluates a system based on knowledge derived from physical, psychological, or sociological theories, theories of design, or usability heuristics. Usability testing refers to the use of empirical investigation conducted with users of the system. Among usability analysis methods, the GOMS family of usability analysis techniques (John and Kieras, 1994) are based on the conceptual framework of human information processing (Newell and Simon, 1972) and aim at predicting the usability of a system before it has been tested. Moreover, GOMS analysis can be used early in the CAAD system development process to evaluate different ideas before the prototypes are implemented.

The general GOMS concept is to analyze knowledge of how to do a task in terms of the components of *goals*, *operators*, *methods*, and *selection rules*. Goals are what a user has to accomplish; they are often decomposed into subgoals, and all of the subgoals must be accomplished in order to achieve the overall goal. Operators are actions performed by the user in service of a goal; they can be perceptual, cognitive, or motor acts, or a composite of these. Methods describe procedures (sequences of operators and subgoals invocations) for accomplishing a goal. Selection rules determine which method to use when there is more than one method available to the user to accomplish a goal.

Figure 1 illustrates a GOMS model with annotations indicating how statements in the model relate to goals, operators, methods and selection rules. This sample model encodes the task of locating an "active node" in a "design space view". Additionally, the GOMS model shows that there are two alternative methods to view the design space: through a "2D tree view" or a "multi-layer view". Lastly, the model shows keystroke-level operations to accomplish a goal. If a designer should choose to view the design space using the multi-layout view, the predicted task performance time could be estimated by adding the times needed to execute all related operators.



```
GOAL: LOCATE-ACTIVE-NODE-IN-DESIGN-SPACE-VIEW
.   GOAL: OPEN-DESIGN-SPACE-VIEW
.   .   [select:
.   .      GOAL: OPEN-2DTREE-VIEW ..if desired 2D tree view
.   .      .   MOVE-POINTER-TO-MENU
.   .      .   CLICK-LEFT-MOUSE-BUTTON
.   .      GOAL: OPEN-MULTILAYER-VIEW ..if desired multi-layer view
.   .      .   MOVE-POINTER-TO-MENU                                   1.10
.   .      .   CLICK-LEFT-MOUSE-BUTTON                                0.20
.   .   ]
.   GOAL: ADJUST-VIEW ..repeat until active node is in view           0.00
                                    TOTAL TIME PREDICTED (SECOND)   1.30
```

Figure 1. A sample GOMS model with time prediction

Constructing a GOMS model is usually time-consuming, especially when the task analysis process is performed as an extra effort aside from the software engineering process. This is where the hybrid approach comes in. In the following sections, I will illustrate how to take advantage of a OOSE use case description and expand it into a GOMS model.

2.2 Object-Oriented Software Engineering: A Use Case Driven Method

Developing a CAAD system is a complex task. OOSE methods attempt to handle the complexity in an organized way by working with models, each of which focuses on a certain aspect of the system. Coyne and his colleagues (1993) demonstrate the use of a particular OOSE method in the development of a design support system called SEED. This practice has enabled SEED researchers to assure the system's functionality through all development phases (for their continuing progress, see SEED, 2000). This particular method—a use case driven OOSE—emphasizes system development based on user requirements (Jacobson, Christerson, et al., 1992).

The core of this use case driven OOSE is a *use case model*. The model specifies a system's functionality from a user's perspective. It uses *actors* and *use cases* to define what exists outside the software system (i.e. actors) and what should be performed by the system (i.e. use cases). The actors represent everything that needs to exchange information with the system. They can be human users or other software systems. However, there is a clear distinction between actors and users—an actor represents only a certain role that a user can play. A use case is a specific way of using the system by an actor through a sequence of interactions to achieve a desired outcome.

For example, in the document of *SEED-Layout Requirement Analysis* (SEED, 1998), each use case is formulated in three parts: a *brief description* that summarizes the use case in a short sentence; a *flow of events* that describes the course of events during the performance of the use case; and a *comments section* for special considerations in the software development process regarding this use case. Using this format, a use case *Show Location of Active Node* can be documented as follows:

> *Show Location of Active Node*
>
> Brief Description:
>
> The system displays a design space view and highlights the active node.
>
> Flow of Events:
>
> 1. The designer requests display of a design space view.
> 2. The system displays a design space view with the active node highlighted.
>
> Comments:
>
> A node represents a *problem* or a *solution*. There may be more than one active node in a design space, i.e. an active problem and its associated active solution. A design space view does not necessarily display a complete view of all nodes populating the design space; it may be a partial view, e.g. a view of the problem decomposition hierarchy only. The highlighting mechanism may employ visual cues in addition to the color change method (e.g. reverse video display) commonly used.

This use case documentation is a base document throughout the software engineering process of a CAAD system. The functional requirement, the user interface design, and even GOMS models for usability evaluation will all be formulated based on this documentation.

2.3 Use Case Driven OOSE Meets GOMS Analysis

The GOMS analysis and the use case driven OOSE share a common point, i.e. both methods operate on a model of user tasks. The GOMS analysis relies on this model (GOMS model) to perform usability evaluations; the OOSE describes system behavior in the use case model, which defines how tasks are performed. Moreover, the GOMS model can be derived from the use case model if a goal-oriented approach (Cockburn, 1997) is taken to develop and structure use cases. My hybrid approach builds on this common base between the GOMS analysis and the use case driven OOSE, and introduces a process that refines the use case model into the GOMS model.

Let us review the use case *Show Location of Active Node* illustrated in Section 2.2. The complete course of events to perform this use case is described in its flow of events section as follows:

Show Location of Active Node

Flow of Events:

1. The designer requests display of a design space view.
2. The system displays a design space view with the active node highlighted.

Considering these two events in terms of user goals during the task, we can define the designer's top level goal:

GOAL: LOCATE-ACTIVE-NODE-IN-DESIGN-SPACE-VIEW

Naturally, if a design space view is available on the screen, the designer can go to that view directly and proceed to look for the active node. Conversely, the designer has to bring up the design space view first if it is not available on the screen. In either cases, the designer may have to adjust the display of the view in order to see the active node. Therefore, we can consider these two events as two subgoals to be achieved in order to accomplish the top level goal. Using GOMS notation, we write:

```
GOAL: LOCATE-ACTIVE-NODE-IN-DESIGN-SPACE-VIEW
. GOAL: OPEN-DESIGN-SPACE-VIEW      ...if not available on screen
. GOAL: ADJUST-VIEW                 ...repeat if active node not in view
```

The indentation above indicates, for instance, that **GOAL:** ADJUST-VIEW is a subgoal of **GOAL:** LOCATE-ACTIVE-NODE-IN-DESIGN-SPACE-VIEW; and the text in italics says that this subgoal is to be invoked repeatedly until the active node appears in view. For comparison, Table 1 illustrates the mapping between the use case event flow and the GOMS model goal hierarchy discussed above.

Table 1. Mapping between use case flow of events and GOMS model goal statements

Show Location of Active Node

Flow of Events	GOMS Model
1. The designer requests display of a design space view.	GOAL: LOCATE-ACTIVE-NODE-IN-DESIGN-SPACE-VIEW . GOAL: OPEN-DESIGN-SPACE-VIEW
2. The system displays a design space view with the active node highlighted.	. GOAL: ADJUST-VIEW ... *repeat if active node not in view*

The completion of GOMS models, however, requires detailed designs of the user interface in order to encode operators, methods, and selection rules. Consequently, the user interface can be evaluated by examining the appropriateness (from the cognitive aspect) of GOMS models and by analyzing data collected from the user testing against GOMS models. For example, as the system development process continues, if a user interface design decision is made to offer the designer two alternative ways to view the design space (e.g. a 2D tree view and a multi-layer view), the subgoal can be expanded to model the choice between two methods:

```
GOAL: OPEN-DESIGN-SPACE-VIEW
. [select:  GOAL: OPEN-2DTREE-VIEW        ...if desired 2D tree view
.           GOAL: OPEN-MULTILAYER-VIEW    ...if desired multi-layer view]
```

The GOMS model can be continuously refined along the software engineering process with little or no extra effort. As more and more user interface design decisions are made (with or without a prototype or actual implementation), the GOMS model will have enough detail to predict the time needed to perform the particular task (e.g., see Figure 1 in Section 2.1).

3. CASE STUDIES

The hybrid approach has been experimented on two projects with promising results. The first project is SEED-Layout (Flemming and Chien, 1995). This approach is used to predict the performance of a newly added functionality. The predictions are compared with the empirical data collected from usability testing. In general, the empirical data agree with the predictions. The second project is a web environment that provides Feng-Shui recommendations for selecting and rearranging (the interior of) an apartment unit (Chien and Shih, 2000). The hybrid approach is used to evaluate user interface designs before the implementation.

3.1 SEED-Layout

SEED-Layout is a generative design system that supports schematic layout planning including the rapid generation of layout alternatives based on explicitly stated requirements (Flemming and Chien, 1995). In a SEED-Layout design session, a designer can explore alternative layout solutions, modify and refine problem specifications, *revisit* promising solutions, and perform many other operations in pursuit of a final layout solution for a design problem. A marker utility is planned and prototyped to provide assistance when a piece of information, such as a promising layout solution, is to be revisited for reasons such as reevaluation or comparison to other solutions. Two methods—GOMS analysis and user testing—are used to assess the marker utility. The two assessment methods are based on the scenario described above: they compare the times a designer may take to visit (first visit) and revisit (second visit) a layout solution in different treatments—without the marker utility or with variations of the utility—of the SEED-Layout environment.

The relevant GOMS models are obtained as a result of the hybrid approach used to develop the marker utility in SEED-Layout (see Appendix A for a sample documentation). This analysis examines the user performance in two tasks:

1. to locate target layouts, and
2. to revisit the target layouts.

The detail GOMS analysis examines user performance in these two tasks in three different SEED-Layout user interface treatments: Control Treatment (without marker utility), Map-and-Notepad Treatment (with markers and marker list), and Map Treatment (with markers only).

The GOMS analysis for the particular task of visiting a specific target layout predicts that having marker support reduces the time a designer takes to revisit that layout. The task performance time could be further reduced if the marker support provides, in addition to visual cues on the display, a marker management utility such as the marker list in one the SEED-Layout user interface treatment.

The three user interface treatments are implemented (see Appendix B for snapshots) and evaluated through empirical studies with two designers to perform the same tasks describe above. A quantitative data analysis based on the keystroke events recorded during each task was performed. The result supports the prediction from the GOMS analysis. (For details of this case study, please refer to Chien, 1998.)

In this case study, the hybrid approach enables SEED-Layout developers to examine the user interface design of a newly planned functionality and predict the performance.

3.2 WIDE-Kindom

Most apartment units in Taiwan are sold before they are ever been built. Therefore, apartment buyers are able to customize the interior of their units (e.g., by changing finishes, fixtures, or layout arrangements) before the building construction begins. WIDE-Kindom is a web environment geared to support this customization process. Among its services, WIDE-Kindom provides "Feng-Shui recommendations" as an alternative to help buyers identify suitable apartment units, as well as fine-tuning apartment layout to achieve a harmonious living environment.

Feng-Shui is a set of rules developed by ancient Chinese to relate people and the man-made environment to the natural environment (for further information regarding Feng-Shui, see Walters, 1991). In Taiwan and Hong Kong, many people arrange their living and working environment according to Feng-Shui. Main entrance, stove, toilet, and bed are four key elements in Feng-Shui. The "location" and "orientation" of each element in a house asserts energy that contributes to either the harmony or the conflict between a resident and the house. On the other hand, people are not all the same; they belong to one of eight different types according to their birthday and time.

WIDE-Kindom developers formulate the use case for providing Feng-Shui recommendations as follows (note that the use case is written from an actor's perspective):

> *Obtain Feng-Shui Recommendations*
>
> Brief Description:
> The system provides Feng-Shui recommendations according to user's birth information and apartment layout.
>
> Flow of Events:
> 1. The user requests Feng-Shui recommendations.
> 2. The system requests input for birth information.
> 3. The system displays apartment layout with recommendations.
>
> Comments:
> This use case should be further elaborated into three supporting use cases: *Input Birth Information*, *Show Available Apartments*, and *Show Feng-Shui Recommendations*.

Throughout the software engineering process, use cases are refined and elaborated. Given the complexity of this task, the developers have noted in the comments section that three supporting use cases should be formulated to accomplish the task. As a result, this forms a hierarchical structure of use cases, which is comparable to the goal hierarchy in a GOMS model. In other words, a GOMS model can be easily formulated as follows:

```
GOAL: OBTAIN-FENG-SHUI-RECOMMENDATIONS
. GOAL: INPUT-BIRTH-INFO
. GOAL: SELECT-APARTMENT
. . GOAL: VIEW-AVAILABLE-APARTMENTS      ...repeat if desired one not in view
. . GOAL: IDENTIFY-APARTMENT
. GOAL: VIEW-RECOMMENDATIONS
. . GOAL: ADJUST-APARTMENT-LAYOUT        ...repeat until satisfied
. . VERIFY-RECOMMENDATIONS
```

As the software engineering process continues, one of the supporting use case *Show Feng-Shui Recommendations* are described as follows:

> *Show Feng-Shui Recommendations*
>
> Brief Description:
> The system displays Feng-Shui recommendations.
>
> Flow of Events:
> 1. The user identifies an apartment layout.
> 2. The system displays the apartment layout with recommendations.
>
> Comments:
> Note that the "recommendations" may be provided in various forms, for example flashing icons to show conflicts or scores for overall layouts.

From here on, WIDE-Kindom developers put their focuses on the user interactions and begin to sketch out possible interface designs. Since the Feng-Shui recommendations service hopes to assist users in customizing the interior of an apartment unit, a decision is make that this service should be interactive and provide recommendations upon each user adjustment. Furthermore, a user adjustment can be made through the four Feng-Shui elements: main entrance, stove, toilet, and bed. Two alternative interaction schemes are discussed: upon the user selection of apartment layout, the system displays an initial recommendation based on the user's birth information; or the user starts with no initial recommendation, and goes on to get the recommendation by adding/adjusting elements on the layout. For comparison, these considerations are encoded in GOMS models shown on Table 2.

Table 2. Two GOMS models for the "View Recommendations" task

Interaction A: no initial recommendation	Interaction B: with initial recommendation
GOAL: VIEW-RECOMMENDATIONS	**GOAL:** VIEW-RECOMMENDATIONS
. **GOAL :** ADJUST-APARTMENT-LAYOUT	. VERIFY-RECOMMENDATIONS
…repeat until satisfied	. **GOAL :** ADJUST-APARTMENT-LAYOUT
. . RECALL-RECOMMENDATIONS	*…repeat if not satisfied*
. . [SELECT	. . RECALL-RECOMMENDATIONS
. . GOAL: ADJUST-MAIN-ENTRANCE	. . [SELECT
. . GOAL: ADJUST-BED	. . GOAL: ADJUST-MAIN-ENTRANCE
. . GOAL: ADJUST-STOVE	. . GOAL: ADJUST-BED
. . GOAL: ADJUST-TOILET]	. . GOAL: ADJUST-STOVE
. . COMPARE-RECOMMENDATIONS	. . GOAL: ADJUST-TOILET]
. VERIFY-RECOMMENDATIONS	. . COMPARE-RECOMMENDATIONS

Having done these analyses, WIDE-Kindom developers implement prototypes with different initial recommendation settings and with varying degrees of freedom for element adjustments (see Appendix C for sample prototypes). The resulting system is publicly accessible and most users appear to enjoy interacting with it.

4. CONCLUSION

The experience gained from two case studies indicates that the hybrid approach enables CAAD system developers to be aware of usability issues and conduct usability evaluation as early as the analysis phase of the software development process. Consequently, this may improve the quality of CAAD software systems as well as ensure the usability of the systems.

For small proof-of-concept prototypes, this approach may not be suitable since conducting direct usability testing may be an easier and more effective method to ensure usability. However, many CAAD software prototypes today are developed with the aim to bring research results closer to practice. To achieve this objective, rigorous methods are needed. The hybrid approach provides just that.

A reviewer commented "[d]ifferent designers will use CAAD systems in different ways, each reflecting the nature of their own design practice." User modeling is difficult. However, through explicitly modeling the behavior of a CAAD system and its users, the CAAD researchers may obtain deeper understanding of the system, the users and interaction between these two (Coyne, Flemming, et al., 1993). The hybrid approach is developed with that in mind. What design drawings and models have done for architects is what the hybrid approach of usability and software engineering hope to do for CAAD system developers.

5. ACKNOWLEDGEMENTS

The SEED-Layout case study was conducted in Carnegie Mellon University with the assistance of the SEED research team, especially Professor Ulrich Flemming. The WIDE-Kindom case study is sponsored by Kindom Construction Co, Taiwan.

6. REFERENCES

Chien, S.-F., 1998, "Supporting Information Navigation in Generative Design Systems", PhD Dissertation, School of Architecture, Carnegie Mellon University, Pittsburgh, PA.

Chien, S.-F. and S.-G. Shih, 2000, "A Web Environment to Support User Participation in the Development of Apartment Buildings", in: *Special Focus Symposium on WWW as the Framework for Collaboration, InterSymp 2000*, July 31-August 5, Baden-Baden, Germany, p.225~231.

Cockburn, A., 1997, "Using Goal-Based Use Cases", *Journal of Object-Oriented Programming*, 10(7), p. 56-62.

Coyne, R.F., U. Flemming, P. Piela, and R. Woodbury, 1993, "Behavior Modeling in Design System Development", in: Flemming and Van Wyk (eds.), *CAAD Futures '93*. Elsevier Science Publishers, Amsterdam, p. 355-354.

Flemming, U. and S.-F. Chien, 1995, "Schematic Layout Design in SEED Environment", *Journal of Architectural Engineering*, 1(4), p. 162-169.

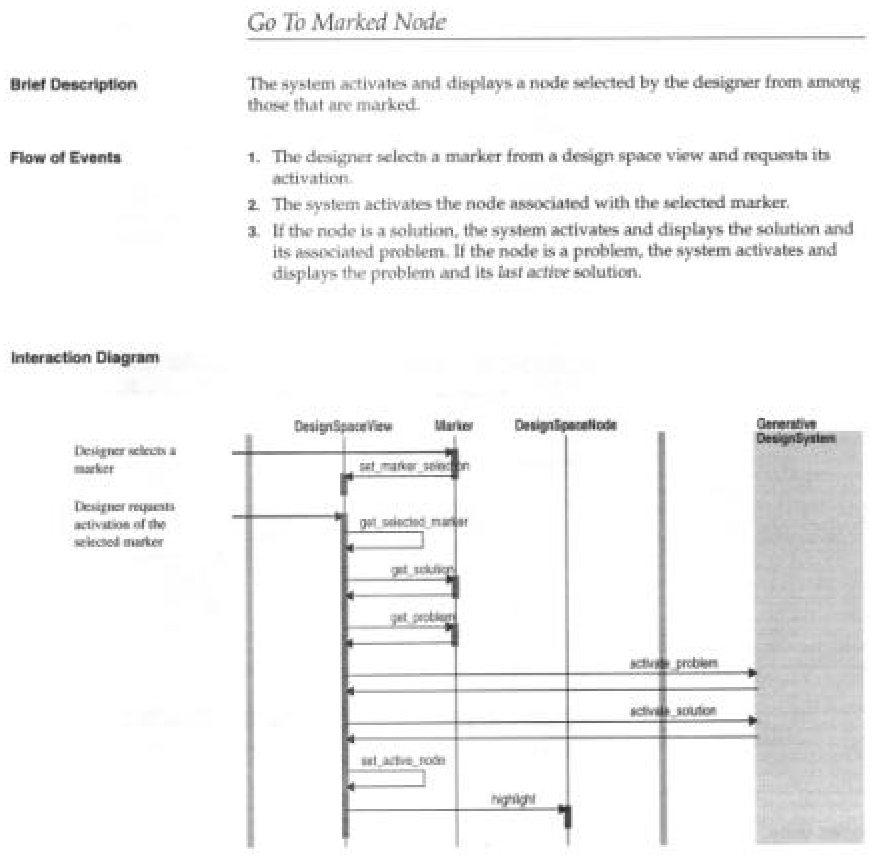
Jacobson, I., M. Christerson, P. Jonesson, and G. Overgaard, 1992, *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley, Reading, MA.

John, B.E. and D.E. Kieras, 1994, "The GOMS Family of Analysis Techniques: Tools for Design and Evaluation", Technical Report CMU-CS-94-181/CMU-HCII-94-106, School of Computer Science, Carnegie Mellon University.

Newell, A. and H.A. Simon, 1972, *Human Problem Solving*, Prentice-Hall, Englewood Cliffs.
Nielsen, J., 1993, *Usability Engineering*, AP Professional, Cambridge, MA.
SEED, 1998, "SEED-Layout Requirement Analysis", URL: http://seed.edrc.cmu.edu/SL/SL-start.book.html.
SEED, 2000, "SEED: Project Web Site", URL: http://seed.edrc.cmu.edu.
Walters, D., 1991, *The Feng-Shui Handbook*, Hopper-Collins, London.

7. APPENDICES

Appendix A: A sample SEED-Layout Requirements Analysis document.



Go To Marked Node

Brief Description

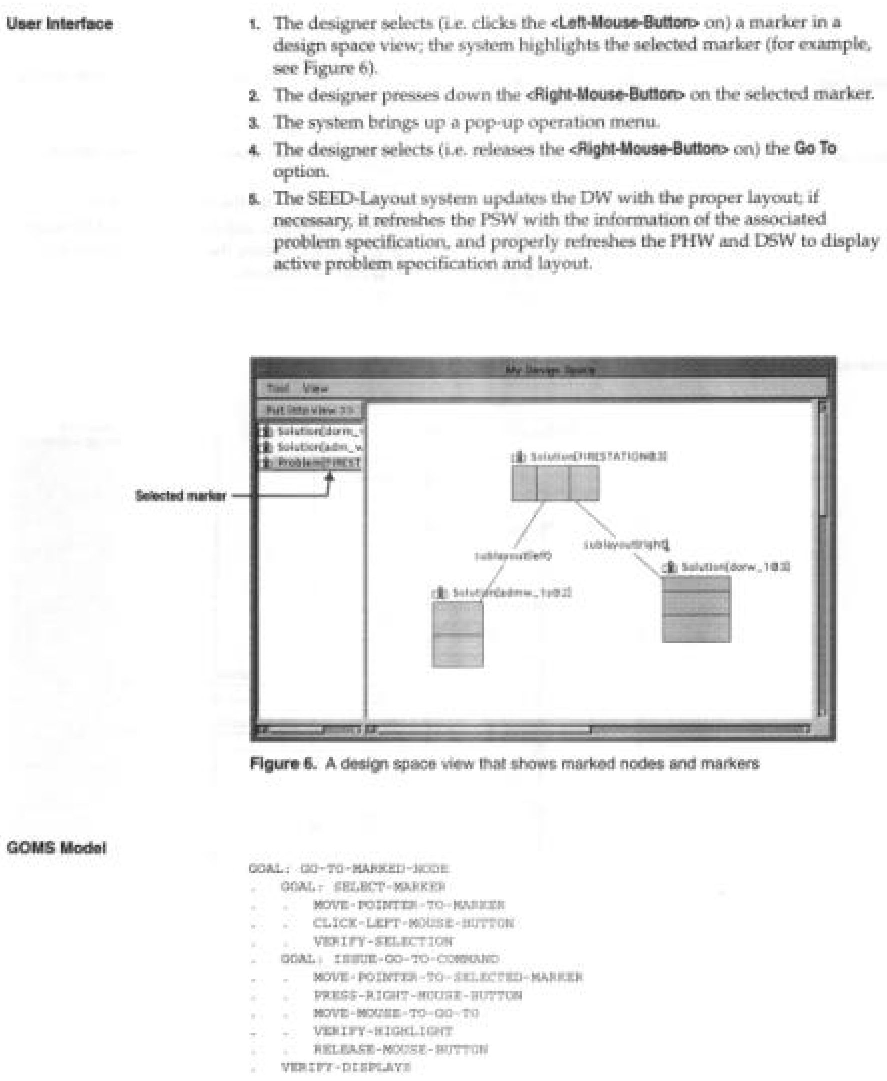
The system activates and displays a node selected by the designer from among those that are marked.

Flow of Events

1. The designer selects a marker from a design space view and requests its activation.
2. The system activates the node associated with the selected marker.
3. If the node is a solution, the system activates and displays the solution and its associated problem. If the node is a problem, the system activates and displays the problem and its *last active* solution.

Interaction Diagram

Figure A-1. Use case document: Go To Marked Node (1)



User Interface

1. The designer selects (i.e. clicks the **<Left-Mouse-Button>** on) a marker in a design space view; the system highlights the selected marker (for example, see Figure 6).
2. The designer presses down the **<Right-Mouse-Button>** on the selected marker.
3. The system brings up a pop-up operation menu.
4. The designer selects (i.e. releases the **<Right-Mouse-Button>** on) the **Go To** option.
5. The SEED-Layout system updates the DW with the proper layout; if necessary, it refreshes the PSW with the information of the associated problem specification, and properly refreshes the PHW and DSW to display active problem specification and layout.

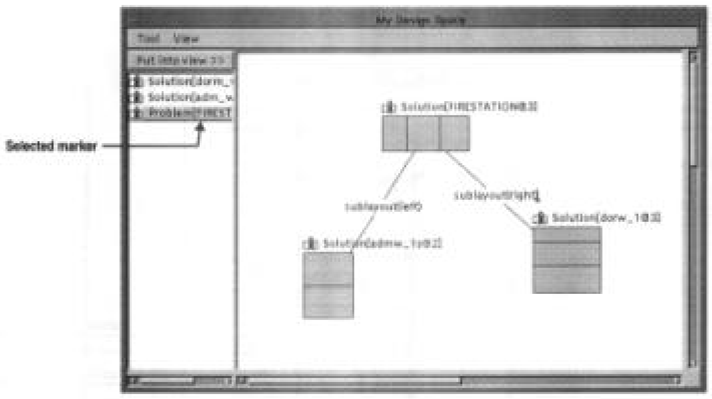


Figure 6. A design space view that shows marked nodes and markers

GOMS Model

```
GOAL: GO-TO-MARKED-NODE
.   GOAL: SELECT-MARKER
.   .   MOVE-POINTER-TO-MARKER
.   .   CLICK-LEFT-MOUSE-BUTTON
.   .   VERIFY-SELECTION
.   GOAL: ISSUE-GO-TO-COMMAND
.   .   MOVE-POINTER-TO-SELECTED-MARKER
.   .   PRESS-RIGHT-MOUSE-BUTTON
.   .   MOVE-MOUSE-TO-GO-TO
.   .   VERIFY-HIGHLIGHT
.   .   RELEASE-MOUSE-BUTTON
.   VERIFY-DISPLAYS
```

Figure A-2. Use case document: Go To Marked Node (2)

Appendix B: Sample screen snapshots for various SEED-Layout user interface treatments.

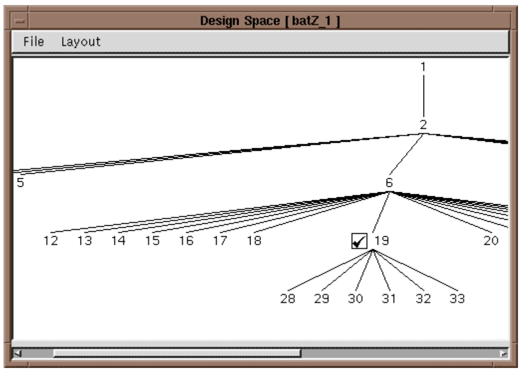


Figure B-1. Control Treatment

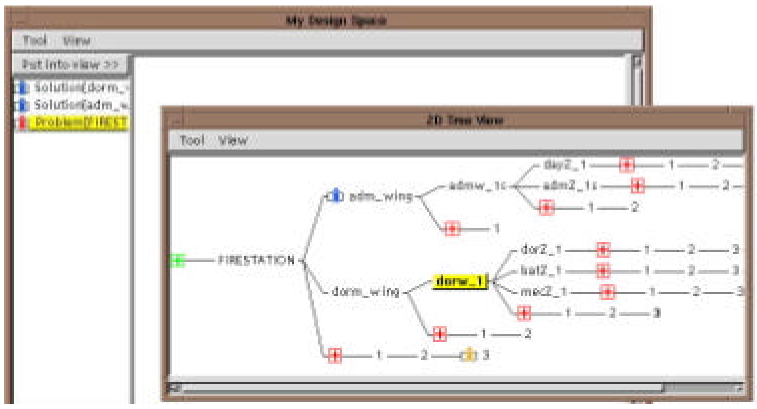


Figure B-2. Map-and-Notepad Treatment

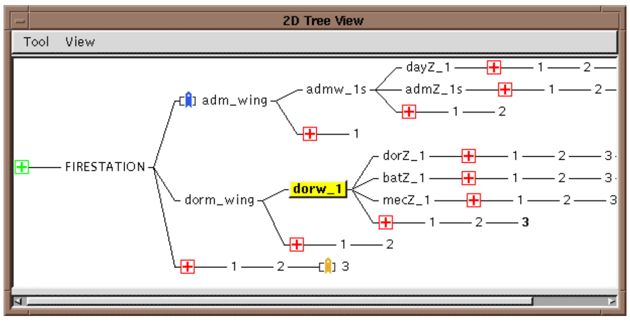


Figure B-3. Map Treatment

Appendix C: Prototypes of WIDE-Kindom Feng-Shui module.

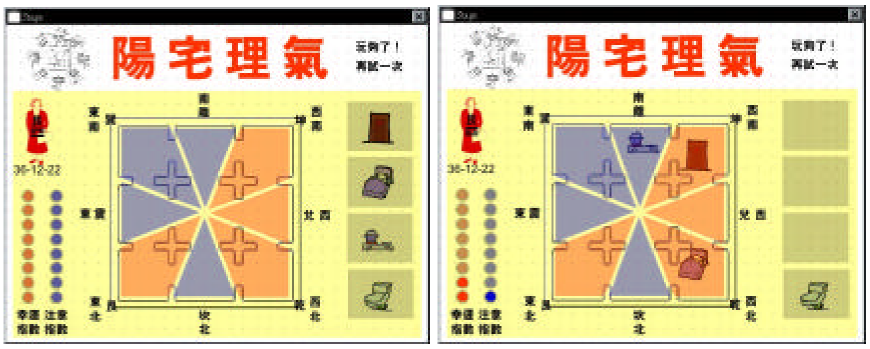


Figure C-1. Version A: no initial recommendations/limited element adjustment

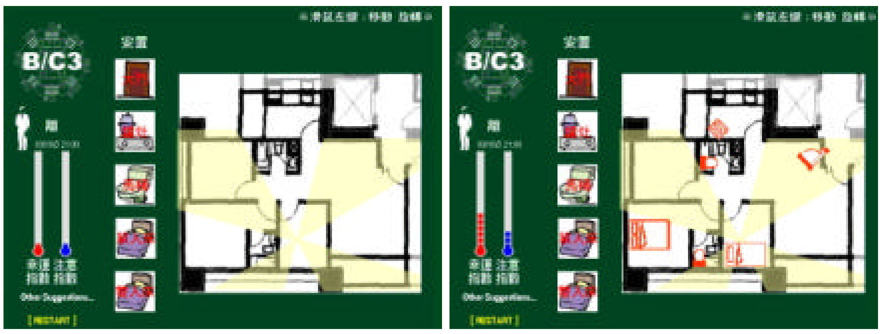


Figure C-2. Version B: no initial recommendations/unlimited element adjustment

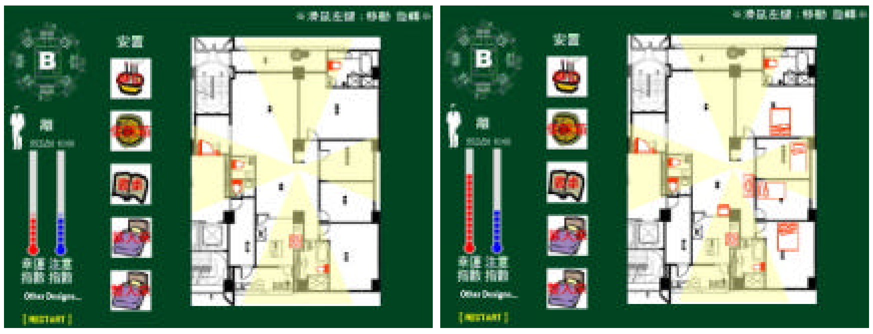


Figure C-3. Version C: initial recommendations/limited element adjustment