Knowledge Elicitation for Design Task Sequencing Knowledge

by

Janet E. Burge

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

By

____

December 1998

APPROVED:

____

Dr. David Brown, Major Advisor

____

Dr. Eva Hudlicka, Adjunct Advisor

____

Dr. Micha Hofri, Head of Department

Abstract

There are many types of knowledge involved in producing a design (the process of specifying a description of an artifact that satisfies a collection of constraints [Brown, 1992]). Of these, one of the most crucial is the design plan: the sequence of steps taken to create the design (or a portion of the design). A number of knowledge elicitation methods can be used to obtain this knowledge from the designer. The success of the elicitation depends on the match between the knowledge elicitation method used and the information being sought. The difficulty with obtaining design plan information is that this information may involve *implicit knowledge*, i.e. knowledge that can not be expressed explicitly.

In this thesis, an approach is used that combines two knowledge elicitation techniques: one direct, to directly request the design steps and their sequence, and one indirect, to refine this knowledge by obtaining steps and sequences that may be implicit. The two techniques used in this thesis were Forward Scenario Simulation (FSS), a technique where the domain expert describes how the procedure followed to solve it, and Card Sort, a technique where the domain expert is asked to sort items (usually entities in the domain) along different attributes.

The Design Ordering Elicitation System (DOES) was built to perform the knowledge elicitation. This system is a web-based system designed to support remote knowledge elicitation: KE performed without the presence of the knowledge engineer. This system was used to administer knowledge elicitation sessions to evaluate the effectiveness of these techniques at obtaining design steps and their sequencing. The results indicate that using an indirect technique together with a direct technique obtains more alternative sequences for the design steps than using the direct technique alone.

Acknowledgements

I would like to thank my advisor, Dr. Dave Brown for the guidance he provided during this thesis. I never would have completed this without his ideas, advice and support. I would also like to thank my co-advisor, Dr. Eva Hudlicka for her help with this thesis. Besides providing her advice and comments on the thesis, she also is the one who got me interested in Knowledge Elicitation and Artificial Intelligence in the first place. She also was instrumental in helping me find a part-time job in order to be able to afford being in graduate school. I would also like to thank my reader, Dr. Carolina Ruiz for reading this thesis, despite my giving it to her two days before she gave birth to her first child. I would also like to thank Dr. Nabil Hachem for helping me choose a domain and task and Dr. Elke Rundensteiner for reviewing my user interface. And, of course, I'd like to thank all my research subjects for taking the time out of their busy schedules to perform my thesis experiment.

In addition, I would like to thank Dr. Greg Zacharias, of Charles River Analytics, for giving me a job that allowed me to disappear for two months while writing my thesis. My friends out in industry are green with envy. I would also like to thank Dr. Karen Lemone, for encouraging me to return to graduate school full time and my parents, for not thinking I was (too) crazy for leaving my full time job. I also want to thank all my friends for supporting me throughout this. You know who you are.

Table of Contents

Table of Figures

List of Tables

Chapter 1 Introduction

There are many types of knowledge involved in producing a design. Of these, one of the most crucial is the design plan: the sequence of steps taken to create the design. In this thesis, an approach is presented that combines two knowledge elicitation techniques to obtain design steps and their sequencing.

1.1 Background

Knowledge Elicitation (KE) is the process of obtaining knowledge from a domain expert that describes how they perform a specific task and/or describes what general knowledge they have about the domain. One use of KE is obtaining knowledge from a person in order to transfer it to a computer program [McGraw & Harbison-Briggs, 1989].

For example, in order to build an expert system to perform medical diagnosis, physicians would be interviewed by a "Knowledge Engineer" to determine what symptoms they look for. The accessibility of this knowledge is dependent on the type of task/knowledge and the subject being questioned. If the task is one that primarily requires motor skills, the chances of the task being performed 'automatically' are much higher than for a task that involves analysis of a problem [Nisbett & DeCamp Wilson, 1977]. Also, different subjects vary in their ability to articulate their knowledge. This is affected both by the skill level of the subject and by how well they are able to verbalize their decisions. The higher the skill level, the more likely it is that the subject will be performing all or parts of the task automatically and will not be able to explain every action [Berry, 1987].

Different KE techniques have been developed in order to obtain knowledge [Boose, 1989], [Cordingley, 1989]. They can be classified along many dimensions. The most common one used is *direct* versus *indirect*, where direct techniques, such as interviewing, are used to obtain information that is easily verbalized, and indirect techniques, such as sorting domain entities into categories, are used to obtain information that is not easily articulated in response to direct questioning [Hudlicka, 1997]. Classification can also be based on the type of interaction with the subject and the type of knowledge most commonly obtained (sequencing, classification, etc.).

This thesis is concerned with the knowledge required to perform design [Brown, 1993]. For design, different types of knowledge are required at different stages of the design process [Smithers, 1998]. Knowledge is required when creating/revising requirements, creating a problem statement, creating a solution or solutions to the problem, and analyzing the results. Design plans [Chandrasekaran, 1990], specifying the actions taken to produce the design, are used to create solutions to the design problem (requirements with a request to design something that satisfies them). These plans require knowing the sequence in which the actions should be taken. The sequence of actions can depend on many factors, including dependencies between subproblems and designer preferences.

Sequencing knowledge can be obtained by using a direct technique such as interviewing. One drawback, however, is that a designer may not be aware of the order in which they perform the steps of their design or why the order is important. Indirect techniques are effective at obtaining information that is less easily expressed. However, indirect techniques are better suited to obtaining information about domain entities and their attributes, not knowledge about process. In this thesis, a combination of direct and indirect techniques is therefore used to overcome this limitation and more effectively elicit the information required to determine the sequencing of design subproblems.

1.2 Problem Description and Motivation

Determining the subproblems in a design plan and a good order in which to perform them is a crucial step in the design process. Sequencing errors will cause the design system to perform unnecessary backtracking or, as a worst case, fail to solve the design problem. In order to determine a correct order the following information is needed from the domain expert:

- The decomposition of the problem,
- The dependencies between the subproblems,
- The degree to which the subproblem solutions are constrained (i.e. the size of the solution space), and
- The order in which the subproblems are normally solved, and why.

Ideally this information would be obtained using a direct technique. Asking the domain expert for exactly the information needed appears to be the most efficient way to get that knowledge. Unfortunately, experts may not be able to readily articulate this knowledge. This could be due to several factors: the expert may have performed the task so often that they are no longer aware of the order of the steps, or they may be aware of the order but not know why (or if) the order is the best one. This suggests that indirect methods may be required to obtain this information.

Indirect techniques are best at classifying information, not identifying process. If only an indirect technique is used, this results in a mismatch between the technique chosen and the information type required. There are several ways to address this problem. One is to modify the method to force it to get the type of information required. For example, repertory grid analysis [Kelly, 1955] could be used to compare plans, rather than domain entities (i.e. the components of the design). Another approach is to use multiple techniques.

1.3 Approach

In order to fully utilize both types of techniques when obtaining sequencing knowledge, a direct technique is used to establish a base of knowledge and an indirect technique is used to identify additional knowledge that is not readily accessible directly. These techniques are combined in an automated KE tool, the Design Ordering Elicitation System (DOES), built as part of this thesis. This tool is administered remotely, so that subjects can participate who are not located at the same location as the knowledge engineer, and is web-based in order to make distribution and administration easier by collecting all of the experiment results in one location. The results of experiments conducted using DOES are analyzed to evaluate the effectiveness of the technique combination.

1.4 Outline

The remaining chapters of this thesis are structured as follows: Chapter 2 discusses existing knowledge elicitation techniques and systems, Chapter 3 discusses design knowledge, Chapter 4 presents the central hypotheses of the thesis, Chapter 5 presents the DOES design, Chapter 6 presents the DOES implementation, Chapter 7 describes the experiments conducted using DOES, Chapter 8 gives the results and evaluation for DOES, and Chapter 9 discusses future research. Sample results from DOES are provided in Appendix B.

Chapter 2 Knowledge Elicitation

Knowledge elicitation (KE) is the process by which knowledge is obtained from the domain expert by the knowledge engineer. In this chapter, different knowledge elicitation techniques and ways that they can be classified are discussed. There is also a brief discussion of existing systems that perform automated knowledge elicitation.

2.1 Knowledge Elicitation Techniques

There are many different knowledge elicitation techniques and it is often difficult to choose between them. Because the success of a knowledge elicitation effort is dependent on the technique chosen, much work has been done to classify knowledge elicitation techniques. In [Cordingley, 1989], KE techniques were grouped into twelve categories. The primary means of classification was the type of interaction with the subject. In [Geiwitz, et al., 1990], an expert system called KATALYST was proposed for selecting the most appropriate knowledge acquisition technique (KAT) for a specific problem.

Three different classification dimensions for KE techniques are discussed here:

- Direct/Indirect,
- Interaction Type, and
- Type of Knowledge Obtained

All of these classifications are important in selecting a KE technique. The choice of direct or indirect can influence how successful the technique might be at capturing knowledge that is not easily expressed. The "interaction type" is a key factor in determining the amount of effort required to administer the technique. The type of knowledge obtained is important so that the technique chosen can be matched to the type of knowledge needed. The choice of technique will determine how easy and effective the knowledge elicitation session will be.

2.1.1 Direct/Indirect

Most KE techniques can be classified as direct or indirect methods. Direct methods are those that obtain the information directly from the expert, i.e., the required information is obtained by asking direct questions or from direct observation. During KE sessions using direct techniques, the expert verbalizes the needed information. Indirect methods are those where the needed information is *not requested directly*. Instead, the results of the knowledge elicitation session must be analyzed in order to extract the needed information. The analysis required depends on the technique and the goals of the knowledge elicitation session.

The advantage of an indirect approach is that these methods can sometimes obtain additional information than that provided by direct methods. There are many reasons why an indirect method might produce more information. One reason is that the indirect method may end up probing aspects of the problem that the knowledge engineer did not anticipate, and did not ask about in the direct KE session. Another reason is that some subjects are not as verbal as other subjects and are unlikely to give full and detailed answers to direct questions. A third reason is that some knowledge may be *implicit*.

Implicit knowledge is knowledge that either was learned implicitly and can not be expressed explicitly, or that was once explicit but has become implicit over time as the domain expert used it repeatedly and it became automatic [Berry, 1987].

2.1.2 Interaction Type

One way of grouping KE methods is to group them by the type of interaction with the domain expert. Table 2-1 shows the categories and the type of information produced. The results listed are typical results; other types of information can also be obtained.

Table 2-1. KE Techniques Grouped by Interaction Type

Category	Technique Examples	Type	Results
Interview	Structured, Unstructured, Semi-Structured	Direct	Varies depending on questions asked
Case Study	Critical Incident Method, Forward Scenario Simulation, Critical Decision Method	Direct	Procedures followed, rationale
Protocols	Protocol Analysis	Direct	Procedures followed, rationale
Critiquing	Critiquing	Direct	Evaluation of problem solving strategy compared to alternatives
Role Playing	Role Playing	Indirect	Procedures, difficulties encountered due to role
Simulation	Simulation, Wizard of Oz	Direct	Procedures followed
Prototyping	Rapid Prototyping Storyboarding	Direct	Evaluation of proposed approach
Teachback	Teachback	Direct	Correction of Misconceptions
Observation	Observation		Procedure followed
Goal Related	Goal Decomposition, Dividing the Domain	Direct	Goals and subgoals, groupings of goals
List Related	Decision Analysis	Direct	Estimate of worth of all decisions for a task
Construct Elicitation	Repertory Grid, Multi-dimensional Scaling	Indirect	Entities, attributes, sometimes relationships
Sorting	Card Sorting	Indirect	Classification of entities (dimension chosen by subject)
Laddering	Laddered Grid	Indirect	Hierarchical map of the task domain
20 Questions	20 Questions	Indirect	Information used to solve problems; organization of problem space

Category	Technique Examples	Type	Results
Document Analysis	Document Analysis	Indirect (usually)	Varies depending on available documents; interaction with experts

The following subsections explain each interaction type and list the applicable KE methods.

2.1.2.1 Interviewing

Interviewing consists of asking the domain expert(s) questions about the domain of interest and how they perform their tasks. Interviews can be unstructured, semi-structured, or structured. The success of an interview session is dependent on the questions asked -- it is difficult to know which questions should be asked, particularly if the interviewer is not familiar with the domain. Its success also depends on the ability of the expert to articulate their knowledge. The expert may not remember exactly how they perform a task, especially if it is one that they perform "automatically".

Some interview methods are used to build a particular type of model of the task. The model is built by the knowledge engineer based on information obtained during the interview and then reviewed with the domain expert. In some cases, the models can be built interactively with the expert, especially if there are software tools available for model creation. Table 2-2 shows a list of interview methods. Again, the output shown is typical. Many of these methods produce raw text from which many different types of information can be extracted.

Table 2-2. Interview Methods

Method	Type	Output	Reference
Interviewing (structured, unstructured, semi-structured)	Direct	Procedures followed, knowledge used (easily verbalized knowledge)	[Hudlicka, 1997], [Geiwitz, et al., 1990]
Concept Mapping	Direct	Concept map	[Hudlicka, 1997], [Thordsen, 1991], [Gowin & Novak, 1984]
Interruption Analysis	Direct	Procedures, problem-solving strategy, rationale	[Hudlicka, 1997]
ARK (ACT-based representation of knowledge) (combination of methods)	Direct	Goal-subgoal network Includes production rules describing goal/subgoal relationship	[Geiwitz, et al., 1990], [Geiwitz, et. al., 1998], [Anderson, 1983]
Cognitive Structure Analysis (CSA)	Direct	Representational format of experts knowledge; content of the knowledge structure	[Geiwitz, et al., 1990], [Leddo&Cohen, 1988]
Problem discussion	Direct	Solution strategies	[Geiwitz, et al., 1990], [Leddo, et. al., 1988]
Tutorial interview	Direct	Whatever expert teaches	[Geiwitz, et al., 1990], [Boose, 1989]
Uncertain information elicitation		Uncertainty about problems	[Geiwitz, et al., 1990], [Boose, 1989]
Data flow modeling	Direct	Data flow diagram (data items and data flow between them – no sequence information)	[OTT, 1998], [Gane & Sarson, 1977]
Entity-relationship modeling	Direct	Entity relationship diagram (entities, attributes, and relationships)	[OTT, 1998], [Swaffield & Knight, 1990]
Entity life modeling	Direct	Entity life cycle diagram (entities and state changes)	[OTT, 1998], [Swaffield & Knight, 1990]
Object oriented modeling	Direct	Network of objects (types, attributes, relations)	[OTT, 1998], [Riekert, 1991]
Semantic nets	Direct	Semantic Net (including relationships between objects)	[OTT, 1998], [Atkinson, 1990]
IDEF (Integrated Definition Language) modeling	Direct	IDEF Model (functional decomposition)	[OTT, 1998], [McNeese & Zaff, 1991]
Petri nets	Direct	Functional task net	[OTT, 1998], [Coovert et al., 1990], [Hura, 1987], [Weingaertner & Lewis, 1988]
Questionnaire	Direct	Sequence of task actions, cause and effect relationships	[OTT, 1998], [Bainbridge, 1979]
Task action mapping	Direct	Decision flow diagram (goals, subgoals, actions)	[OTT, 1998], [Coury et al., 1991]
User Needs Analysis (decision process diagrams)	Direct	Decision process diagrams	[OTT, 1998], [Coury et al., 1991]

2.1.2.2 Case Study

In Case Study methods, different examples of problems/tasks within a domain are discussed. The problems consist of specific cases that can be typical, difficult, or memorable. These cases are used as a context within which directed questions are asked. Table 2-3 shows a list of methods that use cases to obtain information.

Table 2-3. Case Study Methods

Method	Type	Output	Reference
Critical incident strategy	Direct	Complete plan, plus factors that influenced the plan.	[Geiwitz, et al., 1990], [Cordingley, 1989], [Flanagan, 1954]
Forward scenario simulation	Direct	Procedures followed, reasons behind them	[Geiwitz, et al., 1990], [Cordingley, 1989], [Burton&Shadbolt, 1987]
Critical Decision Method	Direct	Goals considered, options generated, situation assessment	[Thordsen, 1991], [Klein et al., 1986]
Retrospective case description	Direct	Procedures used to solve past problems	[Geiwitz, et al., 1990], [Cordingley, 1989]
Interesting cases	Direct	Procedures used to solve unusual problems	[Geiwitz, et al., 1990], [Cordingley, 1989]

2.1.2.3 Protocols

The KE technique called Protocol Analysis [Ericsson and Simon, 1984] involves asking the expert to perform a task while "thinking aloud." The intent is to capture both the actions performed and the mental process used to determine these actions. As with all the direct methods, the success of the protocol analysis depends on the ability of the expert to describe why they are making their decision. In some cases, the expert may not remember why they do things a certain way. In many cases, the verbalized thoughts will only be a subset of the actual knowledge used to perform the task.

One method used to augment this information is Interruption analysis. For this method, the knowledge engineer interrupts the expert at critical points in the task to ask questions about why they performed a particular action.

Table 2-4 lists protocol methods.

Table 2-4. Protocol Methods

Method	**Type**	**Output**	**Reference**
Protocol analysis (think aloud, talk aloud, eidetic reduction, retrospective reporting, behavioral descriptions, playback)	Direct	Procedures, problem-solving strategy	[Hudlicka, 1997], [Ericsson & Simon, 1984], [Geiwitz, et al., 1990]

2.1.2.4 Critiquing

In Critiquing, an approach to the problem/task generated during previous KE sessions is evaluated by the expert. This is used to determine the validity of results of previous KE sessions with the same, or a different, expert. Table 2-5 lists critiquing methods.

Table 2-5. Critiquing Methods

Method	**Type**	**Output**	**Reference**
Critiquing	Direct	Evaluation of a problem solving strategy compared to alternatives	[Geiwitz, et al., 1990], [Cordingley, 1989], [Miller, 1984]

2.1.2.5 Role Playing

In Role Playing, the expert adapts a role and acts out a scenario where their knowledge is used [Cordingly, 1989]. The intent is that by viewing a situation from a different perspective, information will be revealed that was not discussed when the expert was asked directly. For example, if the goal was to find out what information was used in making a medical diagnosis, the expert could be asked to pretend they are interacting with a patient. Table 2-6 shows role playing methods.

Table 2-6. Role Playing Methods

Method	Type	Output	Reference
role playing	Indirect	Procedures, difficulties encountered due to role	[Geiwitz, et al., 1990], [Cordingley, 1989]

2.1.2.6 Simulation

In Simulation methods, the task is simulated using a computer system or other means. This is used when it is not possible to actually perform the task. For example, the design process can be simulated using a multi-agent system to mimic a design team [Shakeri, 1998]. Applying this task to several design projects can be used to determine design methodologies. Table 2-7 shows simulation methods.

Table 2-7. Simulation Methods

Method	Type	Output	Reference
Wizard of Oz	Direct	Procedures followed	[Geiwitz, et al., 1990], [Cordingley, 1989], [Diaper, 1986]
Simulations	Direct	Problem solving strategies, procedures	[Geiwitz, et al., 1990], [Cordingley, 1989]
Problem analysis	Direct	Procedures, rationale	[Geiwitz, et al., 1990], [Leddo, et. al.. 1988]

2.1.2.7 Prototyping

In Prototyping, the expert is asked to evaluate a prototype of the proposed system being developed. This is usually done iteratively as the system is refined. Table 2-8 shows prototyping methods.

Table 2-8. Prototyping Methods

Method	**Type**	**Output**	**Reference**
System refinement	Direct	New test cases for a prototype system	[Geiwitz, et al., 1990], [Leddo, et. al., 1988]
System examination	Direct	Experts opinion on prototype's rules and control structures	[Geiwitz, et al., 1990], [Leddo, et. al., 1988]
System validation	Direct	Outside experts evaluation of cases solved by expert and protocol system	[Geiwitz, et al., 1990], [Leddo, et. al., 1988]
Rapid prototyping	Direct	Evaluation of system/procedure	[Geiwitz, et al., 1990], [Diaper, 1989]
Storyboarding	Direct	Prototype display design	[OTT, 1998], [McNeese & Zaff, 1991]

2.1.2.8 Teachback

In Teachback, the knowledge engineer attempts to teach the information back to the expert, who then provides corrections and fills in gaps. Table 2-9 shows teachback methods.

Table 2-9. Teachback Methods

Method	**Type**	**Output**	**Reference**
teachback	Direct	Correction of misconceptions	[Geiwitz, et al., 1990], [Cordingley, 1989], [Johnson&Johnson, 1987]

2.1.2.9 Observation

In Observation methods, the knowledge engineer observes the expert performing a task. This prevents the knowledge engineer from inadvertently interfering in the process, but does not provide any insight into why the decisions were made. Table 2-10 shows observation methods.

Table 2-10. Observation Methods

Method	**Type**	**Output**	**Reference**
Discourse analysis (observation)	Direct	Taxonomy of tasks/subtasks or functions	[OTT, 1998], [Belkin & Brooks, 1988]
On-site observation	Direct	Procedure, problem solving strategies	[Geiwitz, et al., 1990], [Cordingley, 1989]
Active participation (knowledge engineer interacts with people being observed)	Direct	Knowledge and skills needed for task	[Geiwitz, et al., 1990], [Cordingley, 1989], [Spradley, 1980]

2.1.2.10 Goal Related

In Goal Related methods, focused discussion techniques are used to elicit information about goals and subgoals. For example, in Reclassification, the domain expert is asked to specify what evidence would indicate that a given goal is the correct one [Cordingly, 1989]. Table 2-11 shows goal related methods.

Table 2-11. Goal Related Methods

Method	Type	Output	Reference
Goal Decomposition	Direct	Goals and subgoals	[Geiwitz, et al., 1990], [Breuker & Weilinga, 1983]
Dividing the domain	Direct	How data is grouped to reach a goal	[Geiwitz, et al., 1990], [Cordingley, 1989], [Hart, 1986]
Reclassification	Direct	Evidence needed to prove that a goal was correct	[Geiwitz, et al., 1990], [Cordingley, 1989], [Hart, 1986], [Breuker & Weilinga, 1983]
Distinguishing goals	Direct	Minimal sets of discriminating features	[Geiwitz, et al., 1990], [Cordingley, 1989], [Hart, 1986]
Goal Directed Analysis (goal-means network)	Direct	Goal-means network	[OTT, 1998], [Woods & Hollnagel, 1987]

2.1.2.11 List Related

In List Related methods, the expert is asked to provide lists of information, usually decisions. Table 2-12 shows list related methods.

Table 2-12. List Related Methods

Method	Type	Output	Reference
Decision analysis	Direct	Estimate of worth (value) for all possible decisions for a task	[Geiwitz, et al., 1990], [Cordingley, 1989], [Hart, 1986]

2.1.2.12 Construct Elicitation

So far, all the KE methods discussed have been direct methods. The first classification that includes indirect methods is Construct Elicitation. Construct

Elicitation methods are used to obtain information about how the expert discriminates between entities in the problem domain. The most commonly used construct elimination method is Repertory Grid Analysis [Kelly, 1955]. For this method, the domain expert is presented with a list of domain entities and is asked to describe the similarities and differences between them. These similarities and differences are analyzed to derive the important attributes of the entities. After completing the initial list of attributes, the knowledge engineer works with the domain expert to assign ratings to each entity/attribute pair. The type of rating depends on the information being elicited. In some cases, attributes are rated as present/not present for each entity, in others a scale is used where the attribute is ranked by the degree to which it is present [Hudlicka, 1997]. Table 2-13 shows construct elicitation methods.

Table 2-13. Construct Elicitation Methods

Method	**Type**	**Output**	**Reference**
repertory grid	Indirect	Attributes (and entities if provided by subject)	[Hudlicka, 1997], [Kelly, 1955]
proximity scaling	Indirect	Attributes and relationships	[Hudlicka, 1997]

2.1.2.13 Sorting

In sorting methods, domain entities are sorted to determine how the expert classifies their knowledge. The domain expert is presented with a list of entities to be sorted. They are then asked to sort them either using pre-defined dimensions or along any dimension they feel is important. Subjects may be asked to perform multiple sorts, each using a different dimension. Table 2-14 shows sorting methods.

Table 2-14. Sorting Methods

Method	Type	Output	Reference
card sorting	Indirect	Many types of clustering. (classification)	[Burton & Shadbolt, 1987], [Geiwitz, et al., 1990], [Cordingley, 1989]

2.1.2.14 Laddering

In Laddering, a hierarchical structure of the domain is formed by asking questions designed to move up, down, and across the hierarchy. Examples of concepts would be goals and subgoals, tasks and subtasks. For example, to elicit concepts that are higher in the hierarchy the expert is asked "why" questions. To get concepts lower in the hierarchy the expert is asked "how" questions. To get concepts at the same level, the expert is asked for alternatives [Cordingly, 1989]. Table 2-15 shows laddering methods.

Table 2-15. Laddering Methods

Method	Type	Output	Reference
Laddered grid	Indirect	A hierarchical map of the task domain	[Geiwitz, et al., 1990], [Cordingley, 1989], [Fransella & Bannister, 1977]

2.1.2.15 20 Questions

This is a method used to determine how the expert gathers information. The knowledge engineer selects an item in the problem space (such as a problem or a solution) and the expert questions them to determine what item the knowledge engineer has chosen. Table 2-16 shows the 20 questions method.

Table 2-16. 20 Questions Method

Method	Type	Output	Reference
20 questions	Indirect	Amount and type of information used to solve problems; how problem space is organized, or how expert has represented Task-relevant knowledge.	[Cordingley, 1989], [Geiwitz, et al., 1990], [Spradley, 1979]

2.1.2.16 Document Analysis

Document analysis involves gathering information from existing documentation. This may or may not involve interaction with a human expert to confirm or add to this information. Some document analysis techniques, particularly those that involve a human expert, can be classified as direct. Others, such as collecting artifacts of performance, such as documents or notes, in order to determine how an expert organizes or process information [Cordingly, 1989], are classified as indirect.

Table 2-17 shows documentation analysis methods.

Table 2-17. Document Analysis Methods

Method	Type	Output	Reference
Collect artifacts of task performance	Indirect	How expert organizes or processes task information, how it is compiled to present to others	[Geiwitz, et al., 1990], [Cordingley, 1989]
Document analysis	Direct	Conceptual graph	[OTT, 1998], [Gordon et al., 1993]
Goal Directed Analysis (goal-means network)	Direct	Goal-means network	[OTT, 1998], [Woods & Hollnagel, 1987]

2.1.3 Type of Knowledge Obtained

Besides being grouped into direct and indirect categories, KE methods can also be grouped by the type of knowledge obtained. For example, many of the indirect KE methods are best at obtaining classification knowledge (i.e. the classes that exist in the domain and what entities fit into them) while direct methods are more suited for obtaining procedural knowledge.

Information types used here are:

- Procedures
- Problem solving strategy
- Goals, sub-goals
- Classification
- Relationships
- Evaluation

Many methods fit into more than one category and are listed more than once. Also, this classification shows the information most commonly extracted using a method and does not imply that only that type of information can be elicited.

2.1.3.1 Procedures

Table 2-18 lists methods used to determine the steps followed to complete a task and the order in which they are taken.

Table 2-18. Methods that Elicit Procedures

Method	Category	Output	Type	Reference
Interviewing (structured, unstructured, semi-structured)	Interviewing	Procedures followed, knowledge used	Direct	[Hudlicka, 1997], [Geiwitz, et al., 1990]
Concept Mapping	Interview	Concept Map	Direct	[Hudlicka, 1997], [Thordsen, 1991], [Gowin & Novak, 1984]
Interruption Analysis	Interviewing	Procedures, problem-solving strategy, rationale	Direct	[Hudlicka, 1997]
Problem discussion	Interview	Solution strategies	Direct	[Geiwitz, et al., 1990], [Leddo, et. al., 1988]
Tutorial interview	Interview	Whatever expert teaches!	Direct	[Geiwitz, et al., 1990], [Boose, 1989]
Entity life modeling	Interview	Entity life cycle diagram (entities and state changes)	Direct	[OTT, 1998], [Swaffield & Knight, 1990]
IDEF modeling	Interview	IDEF Model (functional decomposition)	Direct	[OTT, 1998], [McNeese & Zaff, 1991]
Petri nets	Interview	Functional task net	Direct	[OTT, 1998], [Coovert et al., 1990], [Hura, 1987], [Weingaertner & Lewis, 1988]
Questionnaire	Interview	Sequence of task actions, cause and effect relationships	Direct	[OTT, 1998], [Bainbridge, 1979]
Task action mapping	Interview	Decision flow diagram (goals, subgoals, actions)	Direct	[OTT, 1998], [Coury et al., 1991]
Retrospective case description	Case Study	Procedures followed	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989]
Critical incident strategy	Case Study	Complete plan, plus factors that influenced the plan.	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989], [Flanagan, 1954]

Method	**Category**	**Output**	**Type**	**Reference**
Forward scenario simulation	Case Study	Procedures followed, reasons behind them	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989], [Burton & Shadbolt, 1987]
Retrospective case description	Case Study	Procedures used to solve past problems	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989]
Interesting cases	Case Study	Procedures used to solve unusual problems	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989]
protocol analysis (think aloud, talk aloud, retrospective reporting, behavioral descriptions, playback)	Protocols	Procedures, problem-solving strategy	Direct	[Hudlicka, 1997], [Ericsson & Simon, 1984], [Geiwitz, et al., 1990]
teachback	Teachback	Correction of misconceptions	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989], [Johnson & Johnson, 1987]
critiquing	Critiquing	Evaluation of a problem solving strategy compared to alternatives	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989], [Miller, 1984]
role playing	Role Playing	Procedures, difficulties encountered due to role	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989]
Wizard of Oz	Simulation	Procedures followed	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989], [Diaper, 1986]
Simulations	Simulation	Problem solving strategies, procedures	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989]
Problem analysis	Simulation	Procedures, rationale	Direct	[Geiwitz, et al., 1990], [Leddo, et. al., 1988]
On-site observation	Observation	Procedure, problem solving strategies	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989]

2.1.3.2 Problem Solving Strategy

Table 2-19 lists methods that elicit a problem solving strategy. These methods attempt to determine *how* the expert makes their decisions.

Table 2-19. Methods that Elicit Problem Solving Strategy

Method	**Category**	**Output**	**Type**	**Reference**
Interviewing (structured, unstructured, semi-structured)	Interviewing	Procedures followed, knowledge used	Direct	[Hudlicka, 1997], [Geiwitz, et al., 1990]
Interruption Analysis	Interviewing	Procedures, problem-solving strategy, rationale	Direct	[Hudlicka, 1997]
Problem discussion	Interview	Solution strategies	Direct	[Geiwitz, et al., 1990], [Leddo, et. al., 1988]
Tutorial interview	Interview	Whatever expert teaches!	Direct	[Geiwitz, et al., 1990], [Boose, 1989]
Uncertain information elicitation	Interview	Uncertainty about problems	Direct	[Geiwitz, et al., 1990]
Critical incident strategy	Case Study	Complete plan, plus factors that influenced the plan.	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989], [Flanagan, 1954]
Forward scenario simulation	Case Study	Procedures followed, reasons behind them	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989], [Burton & Shadbolt, 1987]
protocol analysis (think aloud, talk aloud, retrospective reporting, behavioral descriptions, playback)	Protocols	Procedures, problem-solving strategy	Direct	[Hudlicka, 1997], [Ericsson & Simon, 1984], [Geiwitz, et al., 1990]

Method	**Category**	**Output**	**Type**	**Reference**
critiquing	Critiquing	Evaluation of a problem solving strategy compared to alternatives	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989], [Miller, 1984]
Wizard of Oz	Simulation	Procedures followed	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989], [Diaper, 1986]
Simulations	Simulation	Problem solving strategies, procedures	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989]
Problem analysis	Simulation	Procedures, rationale (like simulated interruption analysis)	Direct	[Geiwitz, et al., 1990], [Leddo, et. al., 1988]
Reclassification	Goal Related	Evidence needed to prove that a decision was correct	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989]
On-site observation	Observation	Procedure, problem solving strategies	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989]
Goal Directed Analysis (goal-means network)	Interview/Document Analysis	Goal-means network	Direct	[OTT, 1998], [Woods & Hollnagel, 1987]
20 questions	20 Questions	Amount and type of information used to solve problems; how problem space is organized, or how expert has represented Task-relevant knowledge.	Indirect	[Cordingley, 1989], [Geiwitz, et al., 1990], [Spradley, 1979]

2.1.3.3 Goals/Subgoals

Table 2-20 lists methods are used to extract the goals and subgoals (decomposition) that are used when performing a task. These methods are listed separately from procedures since ordering is not necessarily provided.

Table 2-20. Methods that Elicit Goals/Subgoals

Method	Category	Output	Type	Reference
ARK (ACT-based representation of knowledge) (combination of methods)	Interview	Goal-subgoal network Includes production rules describing goal/subgoal relationship	Direct	[Geiwitz, et al., 1990], [Geiwitz, et. al, 1998], [Anderson, 1983]
Task action mapping	Interview	Decision flow diagram (goals, subgoals, actions)	Direct	[OTT, 1998], [Coury et al., 1991]
Critical Decision Method	Case Study	Goals considered, options generated, situation assessment	Direct	[Thordsen, 1991], [Klein et al., 1986]
goal decomposition	Goal Related	Goals and subgoals	Direct	[Geiwitz, et al., 1990], [Breuker & Weilinga, 1983]
Dividing the domain	Goal Related	How data is grouped to reach a goal	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989], [Hart, 1986]
Reclassification	Goal Related	Evidence needed to prove that a decision was correct	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989], [Hart, 1986], [Breuker & Weilinga, 1983]
Distinguishing goals	Goal Related	Minimal sets of discriminating features	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989], [Hart, 1986]
Goal Directed Analysis (goal-means network)	Interview/Document Analysis	Goal-means network	Direct	[OTT, 1998], [Woods & Hollnagel, 1987]

2.1.3.4 Classification

Figure 2-21 lists methods used to classify entities within a domain.

Table 2-21. Methods that Elicit Classification of Domain Entities

Method	Category	Output	Type	Reference
Cognitive Structure Analysis (CSA)	Interview	Representational format of experts knowledge; content of the knowledge structure	Direct	[Geiwitz, et al., 1990], [Leddo&Cohen, 1988]
Data flow modeling	Interview	Data flow diagram (data items and data flow between them – no sequence information)	Direct	[OTT, 1998], [Gane & Sarson, 1977]
Entity-relationship modeling	Interview	Entity relationship diagram (entities, attributes, and relationships)	Direct	[OTT, 1998], [Swaffield & Knight, 1990]
Entity life modeling	Interview	Entity life cycle diagram (entities and state changes)	Direct	[OTT, 1998], [Swaffield & Knight, 1990]
Object oriented modeling	Interview	Network of objects (types, attributes, relations)	Direct	[OTT, 1998], [Riekert, 1991]
Semantic nets	Interview	Semantic Net (inc. relationships between objects)	Direct	[OTT, 1998], [Atkinson, 1990]
Distinguishing goals	Goal Related	Minimal sets of discriminating features	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989]
Decision analysis	List Related	Estimate of worth for all possible decisions for a task	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989]
Discourse analysis (observation)	Observation	Taxonomy of tasks/subtasks or functions	Direct	[OTT, 1998], [Belkin & Brooks, 1988]
Collect artifacts of task performance	Document Analysis	How expert organizes or processes task information, how it is compiled to present to others	Indirect	[Geiwitz, et al., 1990], [Cordingley, 1989]
Document analysis	Document Analysis	Conceptual graph	Direct	[OTT, 1998], [Gordon et al., 1993]
repertory grid	Construct Elicitation	Attributes (and entities if provided by subject)	Indirect	[Hudlicka, 1997], [Kelly, 1955]
multi-dimensional scaling	Construct Elicitation	Attributes and relationships	Indirect	[Hudlicka, 1997]
proximity scaling	Construct Elicitation	Attributes and relationships	Indirect	[Hudlicka, 1997]

Method	**Category**	**Output**	**Type**	**Reference**
card sorting	Sorting	Hierarchical cluster diagram (classification)	Indirect	[Burton & Shardbolt, 1987], [Geiwitz, et al., 1990], [Cordingley, 1989]
laddered grid	Laddering	A hierarchical map of the task domain	Indirect	[Geiwitz, et al., 1990], [Cordingley, 1989], [Fransella & Bannister, 1977]
Ranking augmented conceptual ranking	Other	Conceptual Ranking (ordering by value)	Direct	[OTT, 1998], [Chignell & Peterson, 1988], [Kagel, 1986], [Whaley, 1979]

2.1.3.5 Dependencies/Relationships

Table 2-22 lists methods that obtain relationships between domain entities.

Table 2-22. Methods that Elicit Relationships

Method	Category	Output	Type	Reference
Data flow modeling	Interview	Data flow diagram (data items and data flow between them – no sequence information)	Direct	[OTT, 1998], [Gane & Sarson, 1977]
Entity-relationship modeling	Interview	Entity relationship diagram (entities, attributes, and relationships)	Direct	[OTT, 1998], [Swaffield & Knight, 1990]
Object oriented modeling	Interview	Network of objects (types, attributes, relations)	Direct	[OTT, 1998], [Riekert, 1991]
Semantic nets	Interview	Semantic Net (inc. relationships between objects)	Direct	[OTT, 1998], [Atkinson, 1990]
Questionnaire	Interview	Sequence of task actions, cause and effect relationships	Direct	[OTT, 1998], [Bainbridge, 1979]
Discourse analysis (observation)	Observation	Taxonomy of tasks/subtasks or functions	Direct	[OTT, 1998], [Belkin & Brooks, 1988]
multi-dimensional scaling	Construct Elicitation	Attributes and relationships	Indirect	[Hudlicka, 1997]
Proximity scaling	Construct Elicitation	Attributes and relationships	Indirect	[Hudlicka, 1997]
card sorting	Sorting	Hierarchical cluster diagram (classification)	Indirect	[Burton & Shardbolt, 1987], [Geiwitz, et al., 1990], [Cordingley, 1989]
Laddered grid	Laddering	A hierarchical map of the task domain	Indirect	[Geiwitz, et al., 1990], [Cordingley, 1989], [Fransella & Bannister, 1977]

2.1.3.6 Evaluation

Table 2-23 lists methods that are used to evaluate systems, usually prototype systems, or other types of KE session results.

Table 2-23. Methods that Elicit Evaluations

Method	**Category**	**Output**	**Type**	**Reference**
teachback	Teachback	Correction of misconceptions	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989], [Johnson & Johnson, 1987]
critiquing	Critiquing	Evaluation of a problem solving strategy compared to alternatives	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989], [Miller, 1984]
System refinement	Prototyping	New test cases for a prototype system	Direct	[Geiwitz, et al., 1990], [Leddo, et. al., 1988]
System examination	Prototyping	Experts opinion on prototype's rules and control structures	Direct	[Geiwitz, et al., 1990], [Leddo, et. al., 1988]
System validation	Prototyping	Outside experts evaluation of cases solved by expert and protocol system	Direct	[Geiwitz, et al., 1990], [Leddo, et. al., 1988]
Rapid prototyping	Prototyping	Evaluation of system/procedure	Direct	[Geiwitz, et al., 1990], [Diaper, 1989]
Storyboarding	Prototyping	Prototype display design	Direct	[OTT, 1998], [McNeese & Zaff, 1991]
Decision analysis	List Related	Estimate of worth for all possible decisions for a task	Direct	[Geiwitz, et al., 1990], [Cordingley, 1989]
Ranking augmented conceptual ranking	Other	Conceptual Ranking (ordering by value)	Direct	[OTT, 1998], [Chignell & Peterson, 1988], [Kagel, 1986], [Whaley, 1979]

2.2 Knowledge Elicitation Systems

Performing knowledge elicitation manually has several drawbacks. It is time-consuming for the knowledge engineer and often results in large amounts of unstructured data, needing analysis later. It can often be difficult to be consistent from session to session and subject to subject. To overcome these problems, automated KE tools [Boose, 1989] have been developed.

These KE tools follow several different approaches to knowledge elicitation. Some tools, such as Protégé [Munsen, 1998] and DNA [Shute, 1998], are actually KE tool generators. These tools generate domain-specific KE tools that gather information about their domain.

Other tools, such as TEIRESIAS [Davis, 1979] and ASK [Gruber, 1989] perform knowledge elicitation in the context of performing a specific task. In TEIRESIAS, a program used for KE for a medical diagnosis system, the system interacts with the user to refine errors from an existing knowledge base. ASK follows a similar approach except the goal is to obtain strategic knowledge.

There are also KE systems built to support a specific Problem Solving Method (eg. PSM). TEIRISIAS falls into this category by supporting the Heuristic Classification PSM. MOLE [Eshelman, et. al., 1987] also supports Heuristic Classification and is also used for performing KE for a medical diagnosis system. Another system, SALT [Marcus & McDermott, 1989], a KE tool used to obtain design knowledge, supports the Propose and Revise PSM. The advantage of these tools is that they obtain specific knowledge of interest and structure it in a format that can be used by the expert system, guided by the PSM.

Other tools automate the process of conducting the KE session and recording the results, but are not domain specific or PSM specific. Two examples of this are TOPKAT [Kingston, 1994] and VIEW [Zacharias et al., 1995]. In TOPKAT, several KE techniques (card sort, repertory grid, and laddered grid) are used to obtain classification knowledge in a format similar to that in CommonKADS [Van de Velde, 1994].

In VIEW, the knowledge engineer can create the experiment using many different types of KE methods, either alone or together. The experiment can then be "played" for the domain expert who provides information either by typing it into forms that get stored into a database or by speaking into a microphone and having it recorded on the computer. The advantage of this system is that it is very flexible, the disadvantage is that the data stored is not structured. VIEW is currently being used to conduct computer-assisted KE sessions to obtain data about Military Decision-Making. The system introduced in this thesis, DOES, also automates the KE session and is not domain dependent.

2.3 Related Work

There have been many efforts to classify KE techniques. Cordingly [1989] groups techniques into twelve types: interviewing, focused talk, teach back, construct elicitation, sorting, laddering, '20 questions', matrix generation, critiquing, protocols, role play, and simulations.

Geiwitz, et. al. [1990] designed a conceptual model of the knowledge acquisition process to guide the design of KATalyst, an expert system for choosing an effective knowledge acquisition technique (KAT). They created a list of 150 dimensions for choosing KATS (KE techniques); these dimensions fall into six categories: Domain, Expert, User, Solutions, Errors, and Resources. They also created a catalog of KATS in order to develop a prototype knowledge base for KATalyst.

There have also been several studies comparing different KE techniques. In McCloskey, et. al. [1991], ARK (a structured interview technique) and repertory grid Analysis were compared. The repertory grid technique was successful in obtaining classification knowledge while ARK was successful at obtaining procedural knowledge. When Thordsen [1991] compared concept mapping and critical decision method (CDM), he found that concept mapping captured an overview of the user's image of the task while CDM obtained decisions critical cues, situation assessment and others, including deeper knowledge that separates experts from novices. In Hudlicka [1996], three indirect techniques were compared: repertory grid, multi-dimensional scaling, and hierarchical clustering. Of the three techniques, repertory grid was the easiest to analyze and provided all the attributes produced by the other two methods.

In the next chapter, design knowledge, and its acquisition, is discussed.

Chapter 3 Obtaining Design Plan Knowledge

The KE system developed for this thesis is designed to obtain design plan knowledge, one type of sequencing knowledge. This chapter describes the types of knowledge involved in design, followed by a discussion of design plan knowledge and issues encountered in obtaining it.

3.1 Types of Design Knowledge

In Smithers' model of Engineering design, [Smithers, 1998], the following types of design knowledge are described:

- *Needs and desires*: what the customer wants.
- *Requirements formation knowledge and requirements revision knowledge*: required to write the system requirements.
- *Problem synthesis and specification knowledge, and problem revision knowledge*: required to formulate a problem statement.
- *Problem solving knowledge*: required to generate proposed solutions to the problem.
- *Problem analysis, assessment, and evaluation knowledge*: required to analyze the design solutions.

In addition, design presentation knowledge is needed to keep the customer informed about the progress of each step, and design documentation and rational recovery knowledge is required to document the design.

These knowledge requirements are presented at a high level. What information is available at each step will vary depending on the problem to be solved and how much information is provided by the customer. In some cases, only the needs and desires of the customer are specified, in others the designer may be given a detailed problem specification [Bernaras, 1993]. Many cases fall in between when the designer is presented with initial requirements that may or may not be complete. The following subparagraphs discuss what is involved with each kind of knowledge.

3.1.1 Needs and Desires

This involves a statement of what the customer wants (or thinks he or she wants). The level of detail varies depending on the customer and the problem.

3.1.2 Requirements Formation Knowledge

This consists of additional information needed to turn the needs and desires into actual requirements. In the software world, these are often referred to as "testable requirements." Requirements revision knowledge will also be needed since requirements are likely to require adjustment throughout the process.

Knowledge needed to form requirements includes:

- Description of the artifact or system to be designed;
- Requirements on its behavior;
- Resources required (if applicable).

3.1.3 Problem Specification Knowledge

This is the knowledge needed to transform the requirements into an actual specification of the item being built. Knowledge needed to create a problem specification includes:

- What the design components are (includes information specifying if the component is always required);
- Attributes and possible values for these components;
- Constraints between components;
- Priorities on constraints, information on which constraints can be relaxed;
- Relationships (dependencies) between components;

3.1.4 Problem Solving Knowledge

This involves knowledge required to turn the specification into a solution (or solutions). This consists of a strategy (or perhaps a plan) for how to perform the design and knowledge of what resources are available to build the artifact. Note that resources available is different from the resources required stated in the requirements. In some cases, the client/customer may want specific components involved for various reasons. For example, the company may have a warehouse full of a particular part and would like to reduce their inventory. In other cases, the actual choice of component can be deferred until an exact solution is designed.

Knowledge used to solve the problem, given a specification, includes (not exhaustive):

- Decomposition of the design problem;
- Design subproblem ordering;
- Relationships between subproblems;
- How to recompose the subproblems into a final solution;
- Resources available (such as a part catalog);

The decomposition, ordering, and relationships together create a *design plan*.

3.1.5 Solution Analysis Knowledge

This involves knowledge needed to determine if a given solution meets the requirements. Solution analysis knowledge includes:

- Requirements
- Problem specification
- Solution description
- Rationale for design decisions

3.1.6 Documentation and Rationale Recovery Knowledge

This is knowledge required to both document the process and justify design decisions. This knowledge needs to be captured at each step of the process. Some of this information will be (or should be) a natural output of previous design steps, others will need to be explicitly captured (such as rationale). Documentation and rationale recovery knowledge contains the same information as solution analysis knowledge plus the results of the solution analysis.

3.1.7 Presentation Knowledge

This is the knowledge required to provide feedback to the customer about the design process. How detailed this information should be will depend on the needs of the customer. It should, however, provide the customer with enough information so that they can determine if their needs and desires are being met. The knowledge used will be a subset of that needed for documentation and rationale recovery.

3.2 Design Plan Knowledge

The type of design knowledge discussed in this thesis is the design plan [Chandrasekaran, 1990]. Design plans are a subset of the problem solving knowledge described earlier. The design plans discussed here use a decomposition method/model [Maher, 1990] to perform the design. This involves breaking the problem into subproblems. These subproblems are either solved sequentially or, when possible, in parallel**.** Since subproblems may depend on other subproblems, it is necessary to solve the problems in the (or a) correct order. Otherwise, the system would need to backtrack and make adjustments before coming up with the final solution [Liu & Brown, 1994]. The design plan needs to indicate both the decomposition and the order in which the problems should be solved.

Two factors influence the order in which subproblems should be addressed: the dependencies between subproblems and the number of constraints on a subproblem. If one subproblem depends on the solution to another, they need to be solved in the correct order. If a subproblem is heavily constrained, it makes sense to solve it first. There are two reasons for this: minimizing the amount (and/or length) of backtracking and to ensure that a solution is even possible. A heavily constrained subproblem is likely to have few solutions, hence putting it early in a plan quickly determines whether it has a solution (minimizing backtracking) and reduces the search space in subsequent subproblems. The ordering information (dependencies and constraints) needs to be obtained by some method. This thesis concentrates on obtaining design steps, which are subproblems at a detailed level, and their dependencies. We will consider "steps" to contain enough knowledge to make a single design decision [Brown, 1989].

3.3 Elicitation Challenges

Obtaining design steps and their order could be obtained in a number of ways. The most obvious is to simply ask for them. There are difficulties with this approach. The first difficulty is that the number of possible orders will vary with the level of abstraction used to approach the problem. There may be only one order of the abstract steps, but if broken down into greater detail they may involve steps that can be interleaved to produce several possible orders. The second difficulty is that some of the knowledge used in design may be implicit, i.e. not easily expressed explicitly. The following subsections discuss these issues.

3.3.1 Level of Abstraction

The number of design steps, and the number of orders for these steps, is highly dependent on the level of abstraction of the steps. For example, given the hierarchy in Figure 3-1, the design steps can be ordered at a detailed or a more abstract level.

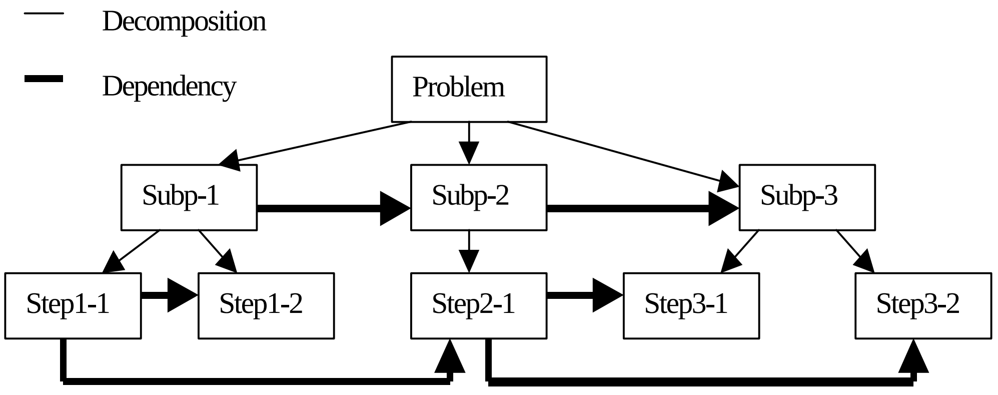


Figure 3-1. Design Step Hierarchy

If the problem was approached at the higher level of abstraction, two orders of the steps would be given: Step1-1->Step1-2->Step2-1->Step3-1->Step3-2 and Step1-1->Step1-2->Step2-1->Step3-2->Step3-1. If the step dependencies are analyzed at the lowest level of abstraction, with only the dependencies remaining fixed, five more orders are found:

- Step1-1->Step2-1->Step1-2->Step3-1->Step3-2
- Step1-1->Step2-1->Step1-2->Step3-2->Step3-1
- Step1-1->Step2-1->Step1-2->Step3-1->Step3-2
- Step1-1->Step2-1->Step3-1->Step3-2->Step1-2
- Step1-1->Step2-1->Step3-2->Step3-1->Step1-2

This suggests that the less abstract the steps are, the more *potential* orders exist. For complex design problems, this would result in large numbers of steps and even larger numbers of potential orders for the steps. For this reason, it is important to either pick a high enough level of abstraction that will allow for a manageable number of design steps for the problem or to determine the number of steps for specific subproblems and not try to interleave them.

3.3.2 Implicit Knowledge

The second issue affecting knowledge elicitation of design steps is the impact of implicit knowledge. It is quite possible that the process of designing may include actions that the designer can not express explicitly, either because they do not view them as something that needs mentioning or because they have become implicit over time. This means that determining steps and their orders is not as simple as just asking for them.

3.4 Related Work

There are many knowledge elicitation methods that could be used for obtaining design knowledge. The method that has been used the most extensively is protocol analysis [Ericsson & Simon, 1984]. It is considered by some [Cross, et. al, 1996] to be the best method for obtaining insight into the design activity. One way of collecting protocol analysis data is videotaping the subjects while they perform the design. This has been done in knowledge elicitation sessions for ADD [Garcia, et. al, 1993], in a study of iteration in a team design task [Smith & Tjandra, 1997] and in knowledge elicitation sessions for the Design History Tool [Chen, et. al., 1990]. In Chen, et al. [1990], it is stated that it takes up to twenty minutes to analyze one minute of protocol data. They were able to decrease this to five minutes with practice.

Very little work was found that was performing knowledge elicitation specifically for design steps and their ordering. Several systems were found, however, that used this type of knowledge. The KADESS system [Price & Kingston, 1993] is based on the KADS methodology [Wielinga, et al., 1993]. The KADESS system is used to support the designer in making safety and stability checks on a building design. The system used a four layer model of expertise: domain, inference, task, and strategy. Design steps and their ordering would be part of the task layer (lower level steps) and strategy layer (order of tasks). PROSUS [Blessing, 1996] is a process-based support system. It uses a design matrix to represent the design process and aid the designers in documenting their design. A strategy matrix was proposed as a means of finding the best way of ordering the design steps but its specification was left as a topic for future research [Blessing, 1994].

One experiment, although not design related, did obtain steps and ordering using a method other than protocol analysis. Kingston & Aitken, [1997] performed knowledge elicitation of intelligence analysis via interviews where high level steps were obtained by asking the domain experts to imagine a textbook teaching intelligence analysis and to then name the book chapters.

In the next chapter, the relationship between implicit knowledge and design plans is discussed.

Chapter 4 Implicit Knowledge & Design Plans

This thesis proposes a process for obtaining design plans that can be used to obtain both implicit and explicit knowledge. This chapter discusses the hypotheses proposed in this thesis, how implicit knowledge is defined for the experiment performed, and the experiment process followed.

4.1 Hypotheses

There are many factors affecting the presence of implicit knowledge in design plans and the success of a KE technique at finding this knowledge. The following hypotheses are examined in this thesis:

- Design plan steps and their order contain implicit knowledge;
- The combination of a direct technique with an indirect technique obtains more information than using the direct technique alone;
- The level of expertise affects the presence of implicit knowledge.;
- The way KE is done (remote vs. in person) affects the quality of the results obtained;
- The amount of implicit knowledge involved depends on the type of domain;
- The amount of implicit knowledge involved depends on the type of task.

4.2 Defining Implicit Knowledge

One difficulty with performing knowledge elicitation specifically to obtain implicit knowledge is determining if something is implicit. This is made harder by the fact that the definition of implicit varies in the literature. By some definitions, implicit knowledge is knowledge that *cannot be expressed* explicitly. However, if some piece of knowledge is not expressed during the knowledge elicitation session but can be derived from post-processing the session results, it may not be knowledge that is implicit by the previous definition (knowledge that cannot be expressed explicitly).

For example, if a domain expert provides a set of design steps and one is found to be missing, is that because the step is implicit or did the domain expert forget to mention it? For the case of design step orderings, if an order is not expressed directly but is derived from the step dependencies, is it necessarily implicit?

Because there is no way to definitively *prove* that a given piece of knowledge is implicit, the concept of "relative implicitness" is introduced here [Brown, 1998]. By this definition, knowledge is viewed as implicit relative to the number of knowledge elicitation sessions conducted where the knowledge is not stated explicitly. If *n* knowledge elicitation sessions are conducted and the knowledge is not stated, the knowledge is rated as implicit at time n, or t_n implicit. As n approaches infinity, the knowledge is more likely to be implicit, for small n, the information may have been forgotten when it was first requested. This measurement indicates the level of implicitness of the knowledge for a particular subject given a specific set of KE techniques administered in a specific order.

4.3 DOES Knowledge Elicitation Process

Indirect KE techniques are often used to obtain implicit knowledge. For the Design Ordering Elicitation System (DOES), the system built for this thesis, the results of a direct technique and an indirect technique will be used to determine if implicit knowledge has been obtained. Figure 4-1 shows how the two techniques are used in DOES.

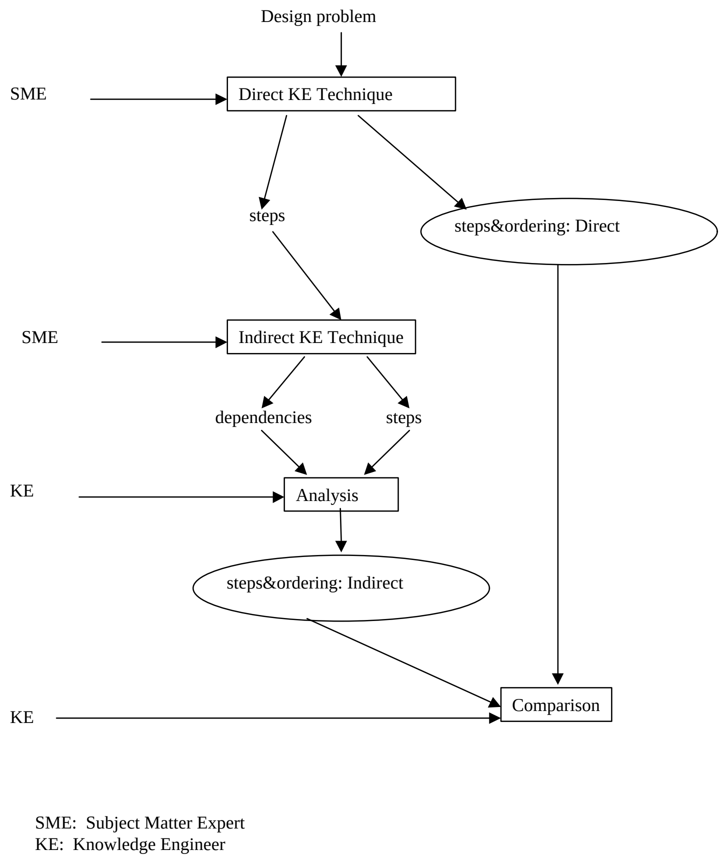


Figure 4-1. DOES Knowledge Elicitation Process

The direct technique will be used to elicit an initial set of design steps and their order from the subject matter expert (SME). The steps are then fed into the indirect technique to obtain additional steps, if any, and dependencies between the steps. The knowledge engineer (KE) then analyses the dependencies to determine additional orders. Steps or order that did not result from the direct KE technique are considered t_1 implicit because they were not obtained during the first technique applied. Some of these might be explicitly stated during the indirect KE. Any additional orders resulting from analysis of the indirect KE technique results are considered t_2 implicit because they were not stated directly by either of the two techniques. There may still be t_n implicit ($n \geq 3$) steps or ordering knowledge that was not obtained during this process.

In the next chapter, the design of DOES is described.

Chapter 5 DOES Design

This chapter discusses the design of DOES, the Design Ordering Elicitation System. This includes the selection of the KE methods implemented, the domain, and the task. The KE process used and the interface design are also discussed.

5.1 KE Method Selection

As described in Section 2, the choice of KE technique used is crucial to the success of the KE effort. The following sections discuss the issues driving technique selection and the techniques chosen.

5.1.1 Issues Driving Selection

There were several issues that drove the selection of an appropriate KE technique for this experiment. The first is the hypothesis that design steps and their ordering involves implicit knowledge. The second is that the technique chosen needs to be easy to administer and analyze. The following sections discuss why the selection was not an easy task.

5.1.1.1 Indirect KE Technique Result Mismatch

In Chapter 2, KE techniques were grouped by the type of knowledge obtained. In Chapter 3, design steps and their ordering were defined as procedural knowledge. In Chapter 4, the hypothesis was presented that design steps and sequencing may involve implicit knowledge.

The KE techniques best suited for procedural knowledge elicitation are all direct techniques. The KE techniques best suited for deriving implicit knowledge are all indirect techniques. Therefore, in order to obtain implicit knowledge that is procedural in nature, it is appropriate to adapt an indirect technique to provide information in the needed form or to combine multiple techniques.

5.1.1.2 Difficulty of Technique

KE techniques vary in how difficult they are to administer. For some, the knowledge engineer needs to be familiar with the domain in order to guide the session in the desired direction. For others, the technique itself requires an experienced knowledge engineer to know which questions should be asked when. Some techniques also rely on the knowledge engineer tailoring their questions based on previous responses; this makes these techniques more difficult to automate and remotely administer (administer with no knowledge engineer present).

5.1.1.3 Scalability

The first indirect technique considered for this experiment was repertory grid analysis [Kelly, 1955]. This is a commonly used indirect technique that has been used successfully in remote KE in the past [Shaw&Gaines, 1995]. The original plan, given in the thesis proposal, was to present the design steps to the subject in groups of three and ask the subjects to indicate the similarities and differences in terms of the information required in order to complete the step. This would have been an interesting approach.

Unfortunately, this method does not scale well. The technique would require that the subject be presented with all possible groups of three steps out of the entered design steps. For example, if the subject entered five steps in the design, they would be asked to compare 10 groups of three steps. If the subject had entered twenty steps in the design, they would be asked to compare 1,140 groups of three steps. Since the subjects are not being forced to limit the number of steps they use, the repertory grid technique is not practical. A better solution would be one where the number of questions presented to the subject was linear with respect to the number of design steps.

5.1.2 Selected Methods

Since direct methods are best for obtaining procedural knowledge and indirect methods are best for obtaining implicit knowledge, a combination of a direct and an indirect method was used for this project. Olsen and Biolsi [1991] suggest that a direct method can be used to determine objects in a domain and an indirect method used to find the relationships between them. In another study, Thordsen [1991] compared Critical Decision Method and Concept Mapping and came to the conclusion that the two techniques could be used very effectively together.

The two techniques chosen for this project were Forward Scenario Simulation (direct) and Card Sort (indirect). Forward Scenario Simulation (FSS) [Burton&Shadbolt, 1987] is a technique where the domain expert is presented with a description of the task and then asked to describe the procedures followed to solve it and the reasons underlying their decisions. This technique was chosen because it does not require that the domain expert be able to perform the task during the KE session or that the knowledge engineer have prior knowledge about how the task is performed. For this experiment, the domain expert was asked to describe how they would perform their design.

Card Sort [Gammack & Young, 1984] is a technique where the domain expert is asked to sort items along different dimensions. For this experiment, the items to be sorted were the design decision steps, and the sort dimension used was their dependencies on prior steps. This method was chosen because it is an indirect method with a high potential for providing multiple orders of steps and because administering it is linear in the number of steps. For this method, twenty steps would involve twenty comparisons, a significant decrease from the 1,140 needed for repertory grid analysis.

5.2 Domain Selection

Many different design domains could have been used for this experiment. The following sections describe the domain chosen and the reasons for that selection.

5.2.1 Issues Driving Selection

There were two primary issues driving the domain selection: the ability of the knowledge engineer (the author) to design the experiment and perform the analysis herself and the availability of research subjects. Because of the limited time available in which to perform the experiments, and to analyze and document the results, it was necessary to choose a domain that was familiar enough to the researcher so that experts from domains other than computer science would not be required to create the experiment and aid in the analysis. It was also necessary to chose a domain where there were many potential research subjects available.

5.2.2 Selected Domain

The domain chosen was database design. The knowledge engineer has been trained in database design and has done some simple design projects. Also, there are a large number of faculty and graduate students in the WPI computer science department available to enlist as research subjects.

5.3 Design Task Selection

After selecting the domain, the next step was to choose a task within the domain. The following sections describe the issues driving the task selection, and the task selected.

5.3.1 Issues Driving Selection

There were several issues driving the selection of the database design task. First, it had to be a frequently performed task so it would be easy to find research subjects who could do it. This ruled out tasks such as query optimization or database tuning which are done by advanced database designers. Second, it needed to be a task where the steps could be performed in more than one order. This suggested that the task should be one that was normally approached at a detailed level. Third, it needed to be a task where some steps were dependent on each other.

5.3.2 Selected Task

The task selected was Entity Relationship Diagram Design. This is a task that is familiar to all, or most, database designers. It is the most critical step in the database design process because it determines what data will be stored in the database and how data items relate to each other. It is also one of the first subjects taught in the graduate level Database Management System course so that even students who had not finished the course could serve as research subjects. This is a task where a given design problem will have multiple solutions and where the steps in a solution could have multiple orders. It is also one where the steps are dependent on each other. For example, it is not possible to determine the relationship between two entities until the entities themselves have been defined.

For this experiment, the subjects were asked to design an ER diagram for an airport database. This database needed to describe entities such as airplanes, airport employees, tests performed on the airplanes, etc. The task stimuli for the experiment are given in Appendix A of this document.

5.4 Remote Knowledge Elicitation

DOES was designed as a Remote Knowledge Elicitation Tool. Remote knowledge elicitation is when the elicitation is performed with the knowledge engineer not being physically present during the experiment. The following sections describe the advantages and disadvantages of this approach.

5.4.1 Advantages

The advantages of remote KE include the following:

- Domain experts can be used who are not collocated with the knowledge engineer.
- Remote KE is cheaper to administer because no travel is involved for the domain experts or the knowledge engineer.
- Data from all experiments can be easily collected in one place.
- Elicitation sessions can be conducted at times convenient to the domain expert.
- Elicitation sessions can be performed in parallel for multiple users because the knowledge engineer does not need to be involved in each one. This reduces the amount of calendar time it takes to perform knowledge elicitation.
- All domain experts are presented with the same stimuli in the same order and results do not depend on their rapport with the knowledge engineer.

5.4.2 Disadvantages

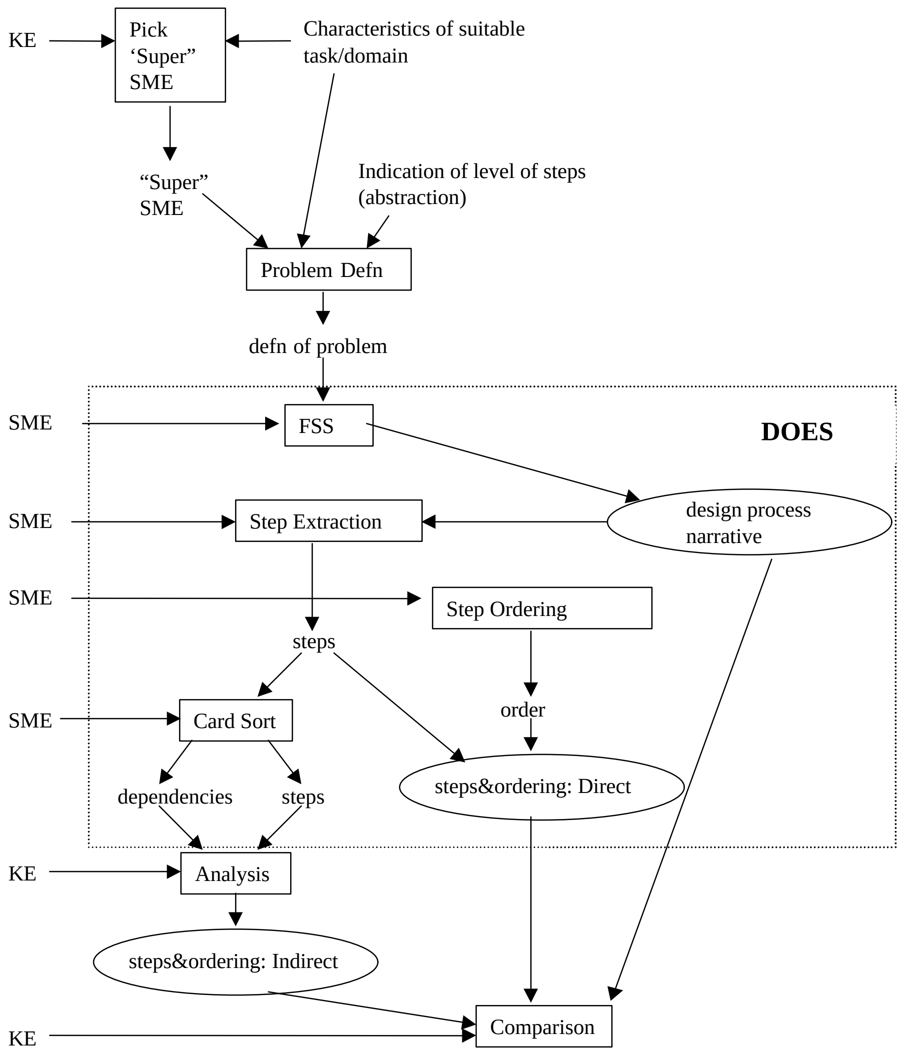
Remote KE has its disadvantages. These include:

- There is no way for the domain expert to ask questions during the experiment.
- It is difficult to predict when the domain experts will perform the experiment.
- KE session results (data produced) are often not as expected since there is no monitoring of how the experiment is performed. Results may be unusable because instructions were misinterpreted or ignored.

- There is the risk that technical difficulties may occur with collection. It is difficult, if not impossible to recover lost data or get the expert to repeat the elicitation session. If the knowledge engineer were present, some of these problems could be prevented.
- The knowledge engineer cannot ask questions that emerge or become relevant during the KE session.

5.5 KE Process

Figure 5-1 shows how the two KE techniques were combined to create the DOES System. The shaded items are part of the DOES implementation. The outputs of the Forward Scenario Simulation (FSS) technique are analyzed and compared with the output of the Card Sort technique. This is done to compare the results obtained using the direct technique alone with the results obtained using the direct technique and the indirect technique combined.



SME: Subject Matter Expert
KE: Knowledge Engineer

Figure 5-1. DOES KE Process

The KE process starts with selecting a domain and a "super Subject Matter Expert" (SME). The "super SME" assists the knowledge engineer in selecting an appropriate domain. The "super SME" for DOES was Dr. Nabil Hachem, of the Worcester Polytechnic Institute Computer Science Department. After the problem has been defined, the DOES system is used to collect information from the research subjects serving as SMEs for the experiment. The subjects first perform the FSS portion of the experiment to create an initial set of steps and their order. They are then instructed to sort the steps based on the dependencies between them. The analysis consists of two parts: using the steps and dependency information to determine the order or orders in which the steps can be performed and comparing the results of the direct technique with the results of both techniques combined.

5.6 Interface Design

The DOES system consists of three parts: a background survey used to obtain information on the level of expertise of the experiment subject, the knowledge elicitation experiment, where the two techniques are used to obtain the design steps and their order, and a usability study to allow the user to evaluate both the design problem and the knowledge elicitation process. Figure 5-2 shows a state transition diagram for the DOES system.

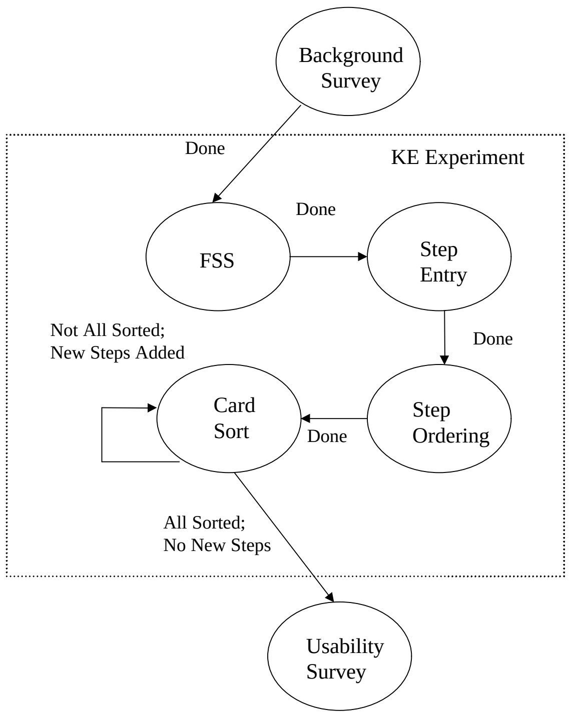


Figure 5-2 DOES System State Diagram

5.6.1 Background Survey

The background survey was created to obtain the following information:

- *Contact information* - This allows the experimenter to keep track of who has performed the experiment. It also provides a way to contact the subjects if there were any questions about the results.
- *Subject design experience* - This includes the educational background of the subject, years of experience in database design, and any relevant training

and/or projects in this area. This information will be used to examine the effect of the subjects' level of expertise on the results.

5.6.2 Knowledge Elicitation Experiment

The knowledge elicitation experiment is where the knowledge elicitation techniques are used to obtain the design steps and their ordering.

5.6.2.1 Design Task Information

The user was given three types of information describing the task:

- *System Requirements* - this information describes the requirements that the design needs to meet.
- *Design Task* - this information describes the specific design task that the subject is to perform.
- *Design Step Example* - this information gives examples of typical design steps for this type of task. This is needed to indicate the level of detail needed in the design solution entered into DOES.

5.6.2.2 DOES Knowledge Elicitation Interface

The knowledge elicitation interface consists of four parts:

- *FSS entry* - this part of the experiment requests that the subject describe ("walk through") the design process.
- *Step entry* - this part of the experiment asks the subject to break the process down into discrete steps.

- *Step sorting* - this part of the experiment asks the subject to take the previously entered steps and sort them into the order in which they were most likely to perform them.
- *Card sort* - this part of the experiment presents the subject with each previously entered step, the "current step", one at a time. For each "current step", the subject is to indicate which of the remaining steps must have been performed before the current step.

5.6.3 Usability Survey

The usability survey was designed to obtain immediate feedback on the DOES system. Information requested includes:

- *Overall usability* - this is used to get the user's overall opinion about the software.
- *Design problem description evaluation* - this is used to determine if the subject had any difficulties solving the problem given the task and problem description.
- *Time to complete the experiment* - this is used to measure how long each subject took to perform the experiment.
- *Evaluation of the approach* - this is used to see if the knowledge elicitation approach helped the user to produce a better design or to better understand how they perform the design.
- *Comments* - this is used to gather any additional feedback from the user.

In the next chapter, the implementation of DOES is described.

Chapter 6   DOES Implementation

This chapter discusses how DOES was implemented.

6.1 Alternatives Considered

After choosing the KE methods and KE process, the next choices that needed to be made were the platform and the programming language. This was done by first examining other remote KE systems. Two systems were studied: DNA [Shute, 1998], and Web Grid II [Shaw & Gaines, 1998]. For DNA, the KE software is given to the domain expert on a floppy disk. They install the program on their computer, perform the experiment, and then store the data on a floppy and return it to the knowledge engineer. For Web Grid II, the KE application consists of a series of HTML forms and is administered via the World Wide Web. Results are stored at a central server.

The advantage of locally installed software is that it does not depend on a computer network. Data are not as likely to get lost or garbled in transmission. Since the environment is more stable, it is easier to test the software. There are several disadvantages to this approach. One disadvantage is that distributing the software and data could be time consuming depending on where the domain expert is located relative to the knowledge engineer and how prompt the domain expert is about sending back the results once the experiment has been completed. Another disadvantage is that the domain expert has to perform the software installation him or herself. This adds to the amount of work they have to perform. A third disadvantage is that the knowledge engineer will then need to manage data from multiple experts received on multiple floppies.

There are several advantages to a web-based approach. The first is that it offers fast turn-around for administering the experiment. All the domain expert needs to perform the experiment is the URL. This can be e-mailed to them and will avoid the delay of distributing experimental materials. Also, the experiment results are stored in a central location immediately upon completion of the experiment. This allows management of the results to be built into the experiment rather than being performed manually by the knowledge engineer. Another advantage is that web-browsers run on multiple platforms -- this means that the domain expert is not required to use a specific machine to perform the experiment. There are also disadvantages. One disadvantage is that it requires the network to be reliable. Another is that the software might not perform consistently across platforms and web-browsers. The web-based approach was chosen for DOES because it is much easier to administer and does not involve the time delays involved in distributing the materials and receiving the results.

The next decision was to choose how to implement the application. The two options considered were HTML forms and Java. HTML forms are easy to implement and behave consistently across platforms. Java is not as easy to implement and behaves differently depending on the version of the web-browser used. It does, however, have the advantage of offering more dynamic displays and more control over user actions than

HTML. For these reasons, Java was chosen as the primary implementation language for DOES.

6.2 DOES Implementation

DOES was implemented in a combination of HTML forms, Java, and Perl. It was designed to be design task independent, i.e. a different design problem could be substituted without modifying any code. It was also designed to be as platform independent as possible. This was done by using Perl, JavaScript, Java, and HTML forms, rather than a compiled language.

Figure 6-1 shows the DOES System Architecture.

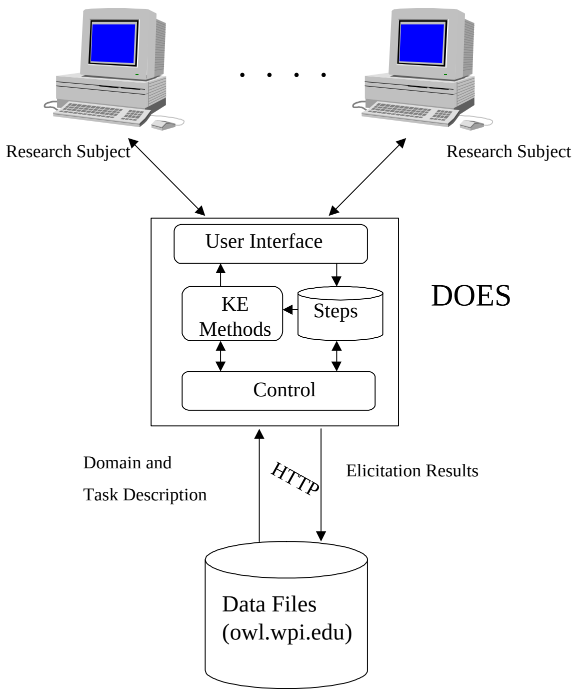
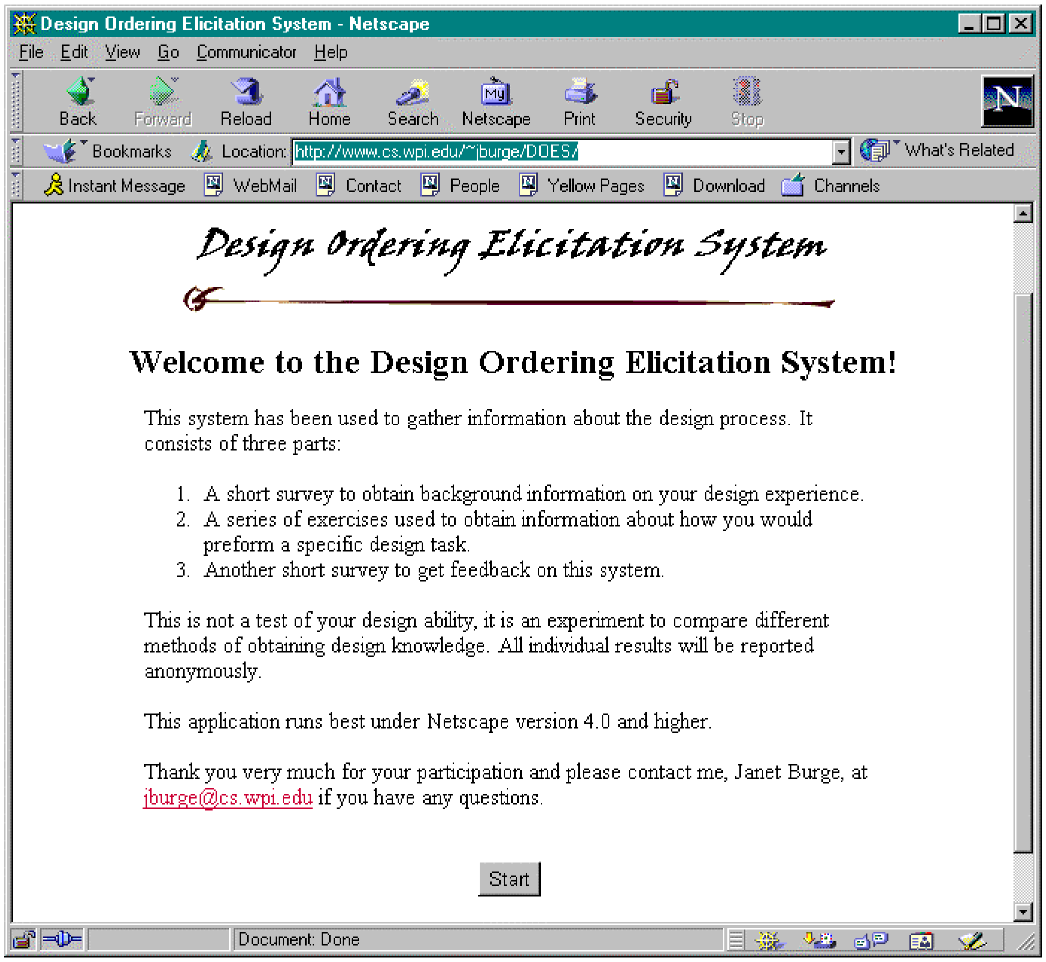


Figure 6-1. DOES System Architecture

6.2.1 Experiment Introduction Display

The experiment began by presenting the subject with a brief description of the three major parts of the experiment: the background survey, knowledge elicitation experiment, and usability survey. They were also provided with contact information in case they had any questions. This page was an HTML form with a "Start" button at the bottom that invoked a Perl script to continue the experiment. Figure 6-2 shows the Experiment Introduction Display.


Design Ordering Elicitation System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://www.cs.wpi.edu/~jburge/DOES/ What's Related
Instant Message WebMail Contact People Yellow Pages Download Channels
Start
Document: Done

Design Ordering Elicitation System

Welcome to the Design Ordering Elicitation System!

This system has been used to gather information about the design process. It consists of three parts:

1. A short survey to obtain background information on your design experience.
2. A series of exercises used to obtain information about how you would preform a specific design task.
3. Another short survey to get feedback on this system.

This is not a test of your design ability, it is an experiment to compare different methods of obtaining design knowledge. All individual results will be reported anonymously.

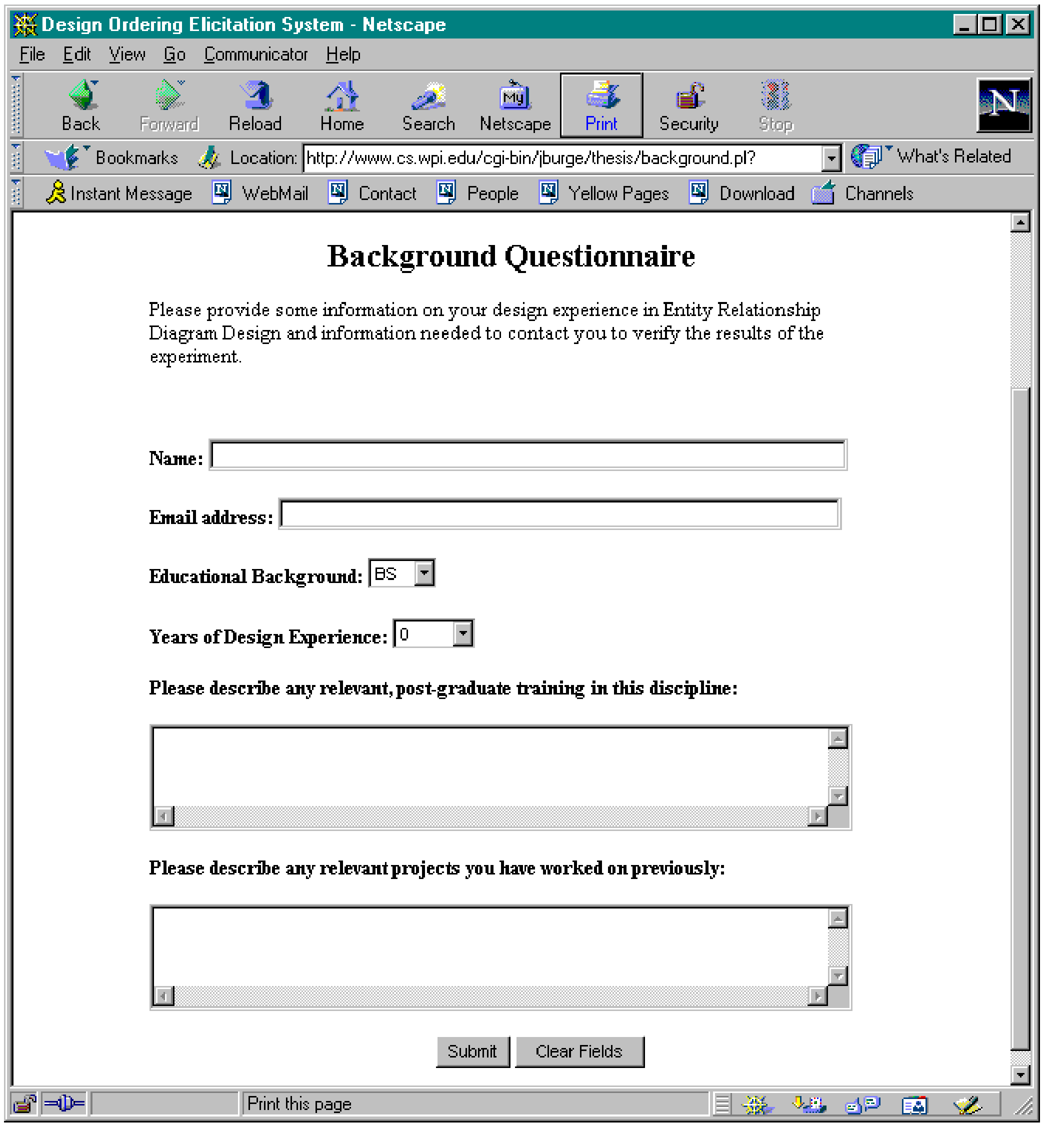
This application runs best under Netscape version 4.0 and higher.

Thank you very much for your participation and please contact me, Janet Burge, at jburge@cs.wpi.edu if you have any questions.

Figure 6-2. Experiment Introduction Display

6.2.2 Background Survey Display

The background survey was written using a Perl script. This script creates the HTML forms used for the survey questions. The Perl script adds the type of design task being performed to the survey introduction so the user can base their answers on the exact task and domain. Figure 6-3 shows the Background Survey Display.


Design Ordering Elicitation System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://www.cs.wpi.edu/cgi-bin/jburge/thesis/background.pl?
What's Related
Instant Message WebMail Contact People Yellow Pages Download Channels
Submit Clear Fields
Print this page

Background Questionnaire

Please provide some information on your design experience in Entity Relationship Diagram Design and information needed to contact you to verify the results of the experiment.

Name:

Email address:

Educational Background: BS

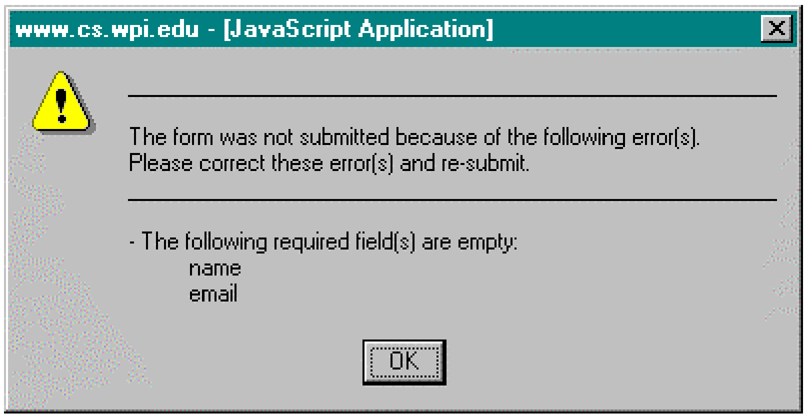
Years of Design Experience: 0

Please describe any relevant, post-graduate training in this discipline:

Please describe any relevant projects you have worked on previously:

Figure 6-3. Background Survey Display

JavaScript is used to check that the name and e-mail fields have been entered and that the e-mail field is a full address (i.e. including the "@" character). If there are errors, the user is informed and they can not submit the form until corrections have been made. The JavaScript code for this was based on an example given in [Flanagan, 1997a]. Figure 6-3 shows the error message displayed.



www.cs.wpi.edu - [JavaScript Application]

The form was not submitted because of the following error(s).
Please correct these error(s) and re-submit.

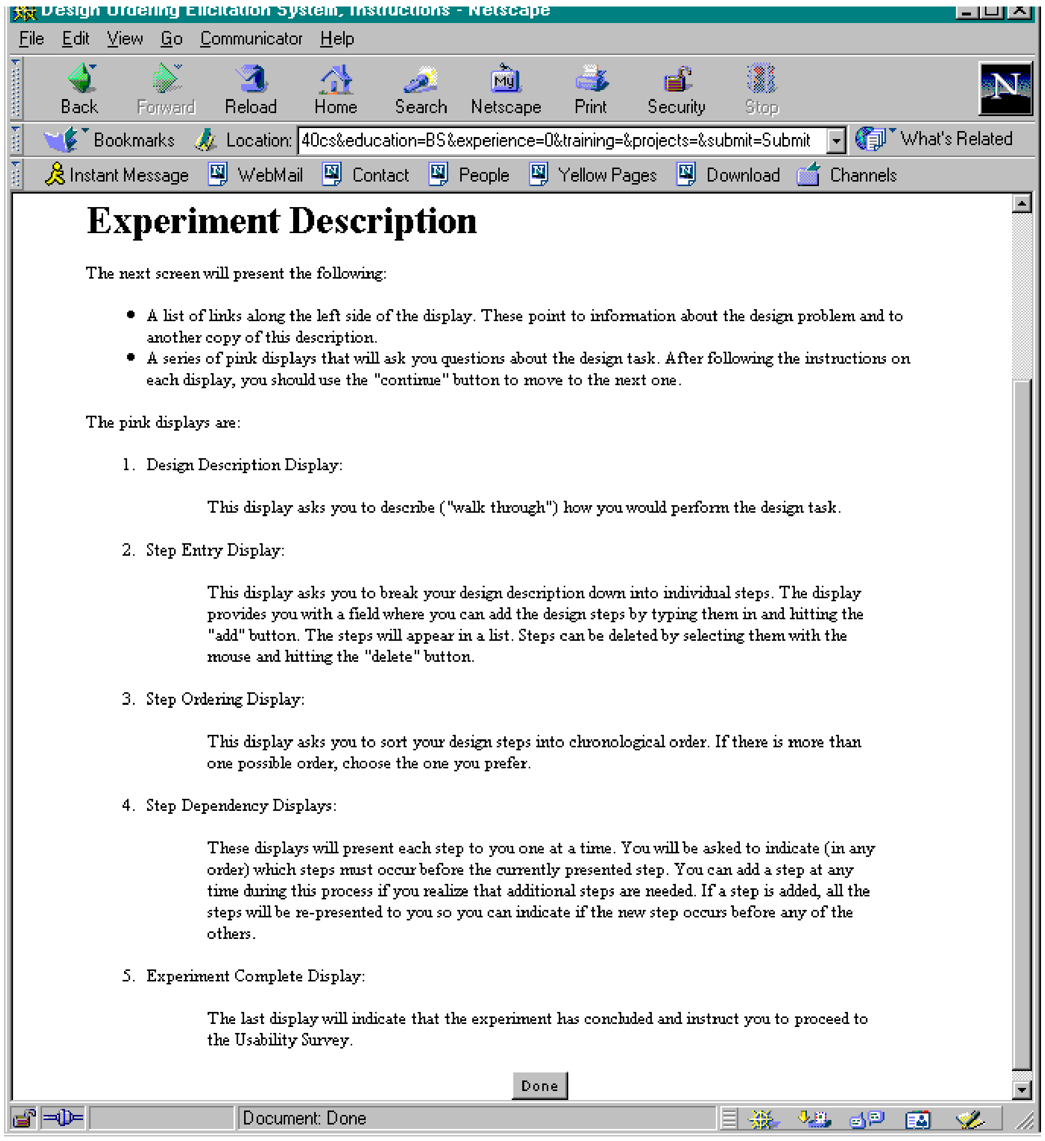
- The following required field(s) are empty:
    name
    email

Figure 6-3. Background Survey Display

When the form is submitted, the results are stored in a text file. The user ID is extracted from the e-mail address and "_background.dat" is appended to create the file name.

6.2.3 DOES Instructions Display

After completing the background survey, a set of instructions is displayed. These instructions explain the remainder of the experiment. Figure 6-4 shows the DOES Instructions Display.


Design Ordering Elicitation System, Instructions - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: 40cs&education=BS&experience=0&training=&projects=&submit=Submit What's Related
Instant Message WebMail Contact People Yellow Pages Download Channels
Done
Document: Done

Experiment Description

The next screen will present the following:

- A list of links along the left side of the display. These point to information about the design problem and to another copy of this description.
- A series of pink displays that will ask you questions about the design task. After following the instructions on each display, you should use the "continue" button to move to the next one.

The pink displays are:

1. Design Description Display:

   This display asks you to describe ("walk through") how you would perform the design task.

2. Step Entry Display:

   This display asks you to break your design description down into individual steps. The display provides you with a field where you can add the design steps by typing them in and hitting the "add" button. The steps will appear in a list. Steps can be deleted by selecting them with the mouse and hitting the "delete" button.

3. Step Ordering Display:

   This display asks you to sort your design steps into chronological order. If there is more than one possible order, choose the one you prefer.

4. Step Dependency Displays:

   These displays will present each step to you one at a time. You will be asked to indicate (in any order) which steps must occur before the currently presented step. You can add a step at any time during this process if you realize that additional steps are needed. If a step is added, all the steps will be re-presented to you so you can indicate if the new step occurs before any of the others.

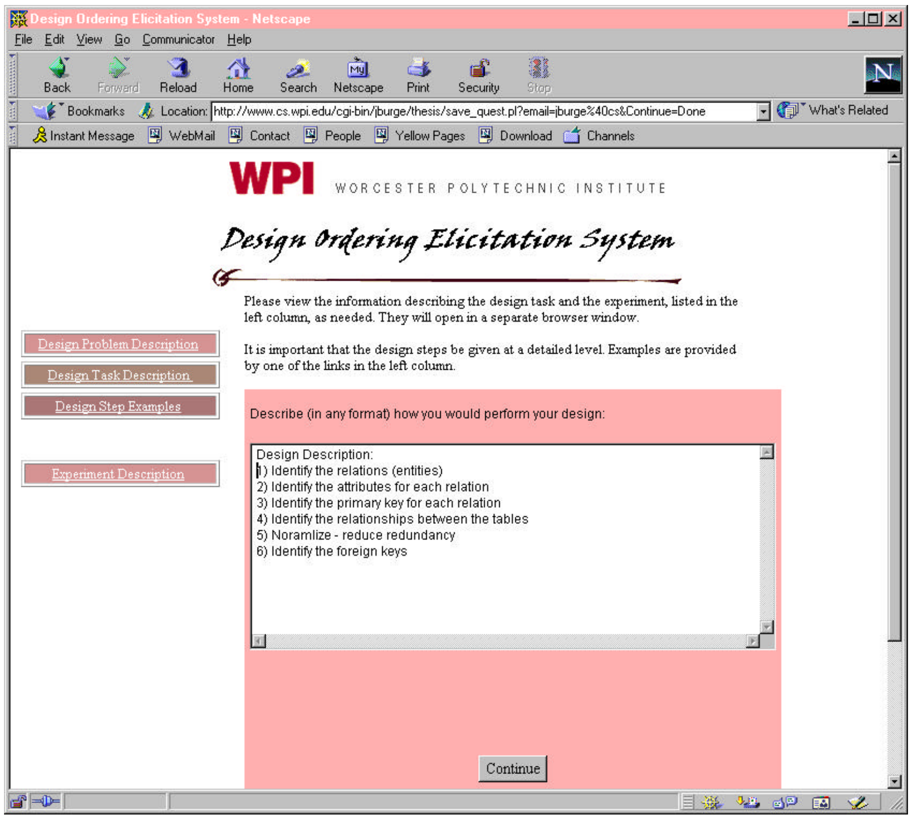
5. Experiment Complete Display:

   The last display will indicate that the experiment has concluded and instruct you to proceed to the Usability Survey.

Figure 6-4. DOES Instructions Display

6.2.4 DOES Knowledge Elicitation Experiment

The DOES knowledge elicitation experiment is implemented using a Perl script that creates an HTML form with an embedded Java applet. The form contains links to the experiment stimuli, the applet, and a button that transitions from the knowledge elicitation experiment to the usability study. The experiment stimuli links are generated from a text file, shown in Appendix A, that contains domain and task specific stimuli. If a different domain or task is used, the file can simply be replaced or modified. The Java applet contains no domain or task specific information Figure 6-5 shows the initial DOES Knowledge Elicitation Experiment Display.


Design Ordering Elicitation System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location:
What's Related
Instant Message WebMail Contact People Yellow Pages Download Channels
Continue

Location: http://www.cs.wpi.edu/cgi-bin/jburge/thesis/save_quest.pl?email=jburge%40cs&Continue=Done

WPI WORCESTER POLYTECHNIC INSTITUTE

Design Ordering Elicitation System

Please view the information describing the design task and the experiment, listed in the left column, as needed. They will open in a separate browser window.

It is important that the design steps be given at a detailed level. Examples are provided by one of the links in the left column.

Design Problem Description

Design Task Description

Design Step Examples

Experiment Description

Describe (in any format) how you would perform your design:

```
Design Description:
1) Identify the relations (entities)
2) Identify the attributes for each relation
3) Identify the primary key for each relation
4) Identify the relationships between the tables
5) Noramlize - reduce redundancy
6) Identify the foreign keys
```

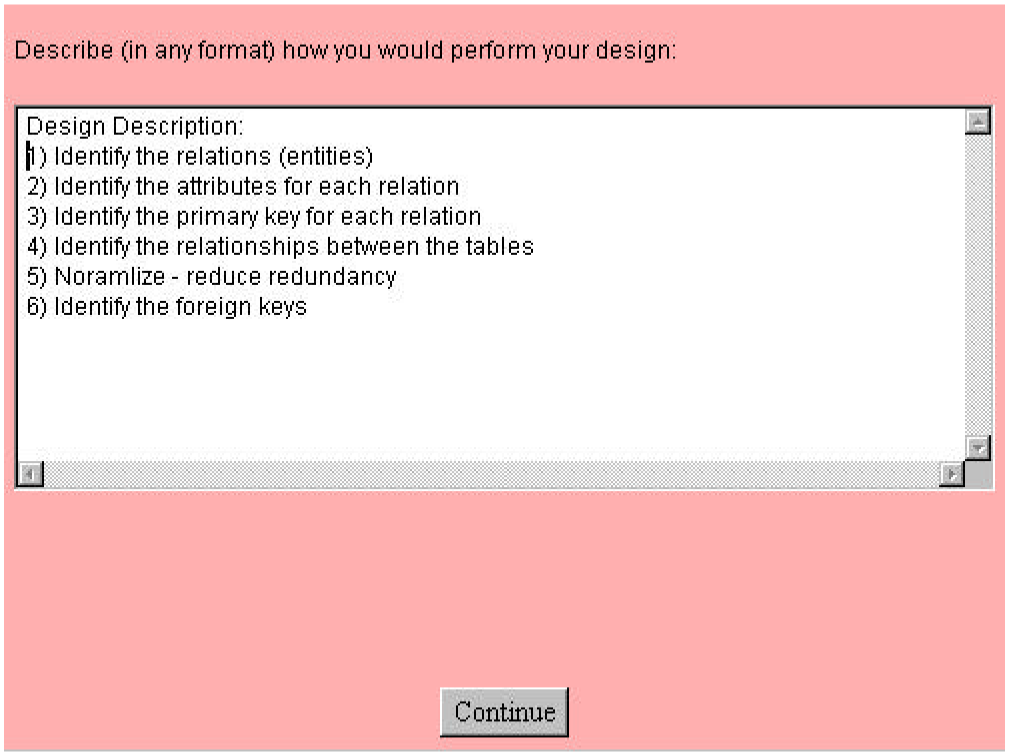
Figure 6-5. Initial DOES Knowledge Elicitation Experiment Display

The following sub-sections describe each of the knowledge elicitation displays. These are all written in Java with some Perl scripts used for file I/O. Code used to lay out some of the displays was modified from [Flanagan, 1997b].

6.2.4.1 Forward Scenario Simulation Display

The first display in the knowledge elicitation experiment uses the Forward Scenario Simulation method. For this method, the subject is presented with a text entry area and is asked to describe how they would perform the design task, entering the actions taken in any format. After this has been completed, the Java applet stores the results in a file by invoking a Perl script via HTTP. The string "_knowledge.dat" is appended to the subject's user ID (created earlier) to create the file name for the "knowledge file".

Figure 6-6 shows the Forward Scenario Simulation Display.



Describe (in any format) how you would perform your design:

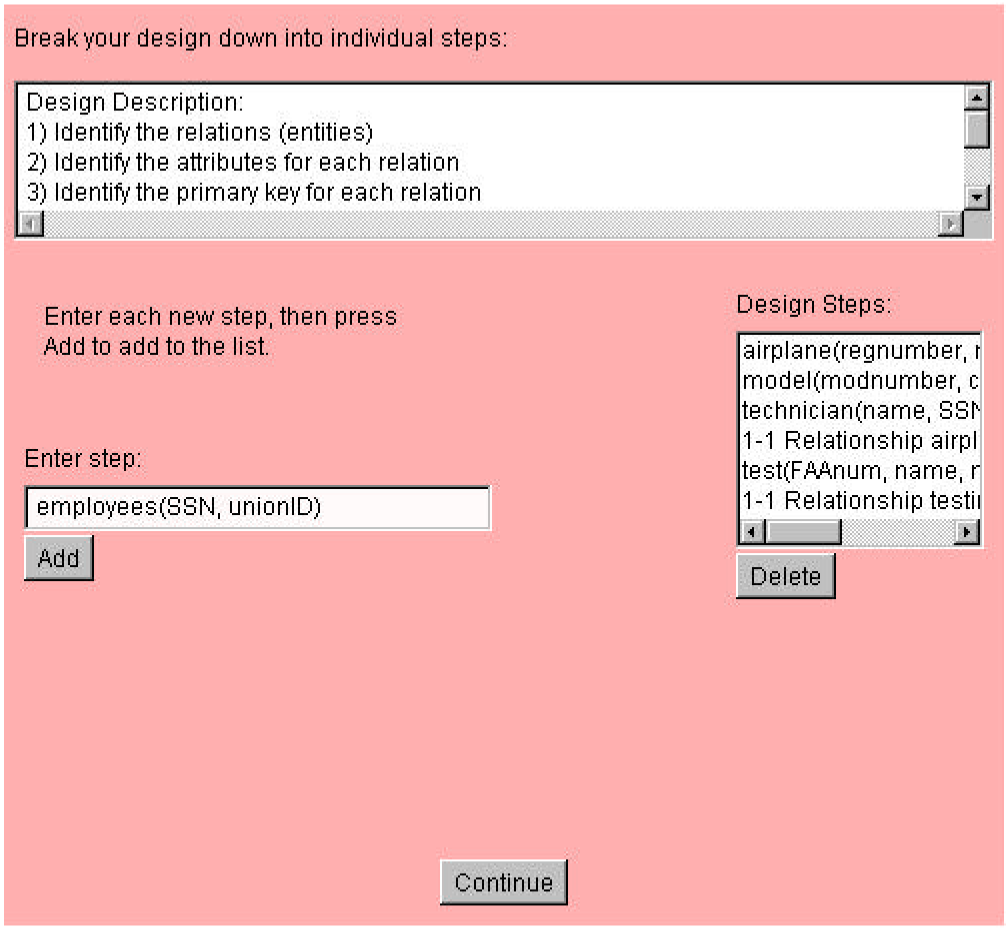
Design Description:
1) Identify the relations (entities)
2) Identify the attributes for each relation
3) Identify the primary key for each relation
4) Identify the relationships between the tables
5) Noramlize - reduce redundancy
6) Identify the foreign keys

Figure 6-6. Forward Scenario Simulation Display

6.2.4.2 Step Entry Display

After the subject has described their design process in free text, they are asked to break it down into discrete steps. They are presented with a non-editable display of their original description and then given a text box that they can use to enter the steps. As they add each step, it is displayed in a list. They can delete steps from the list by selecting them and using a "delete" button. The final list of steps is appended to the knowledge file created earlier.

Figure 6-7 shows the Step Entry Display.


Break your design down into individual steps:

Design Description:
1) Identify the relations (entities)
2) Identify the attributes for each relation
3) Identify the primary key for each relation

Enter each new step, then press Add to add to the list.

Enter step:

employees(SSN, unionID)

Design Steps:

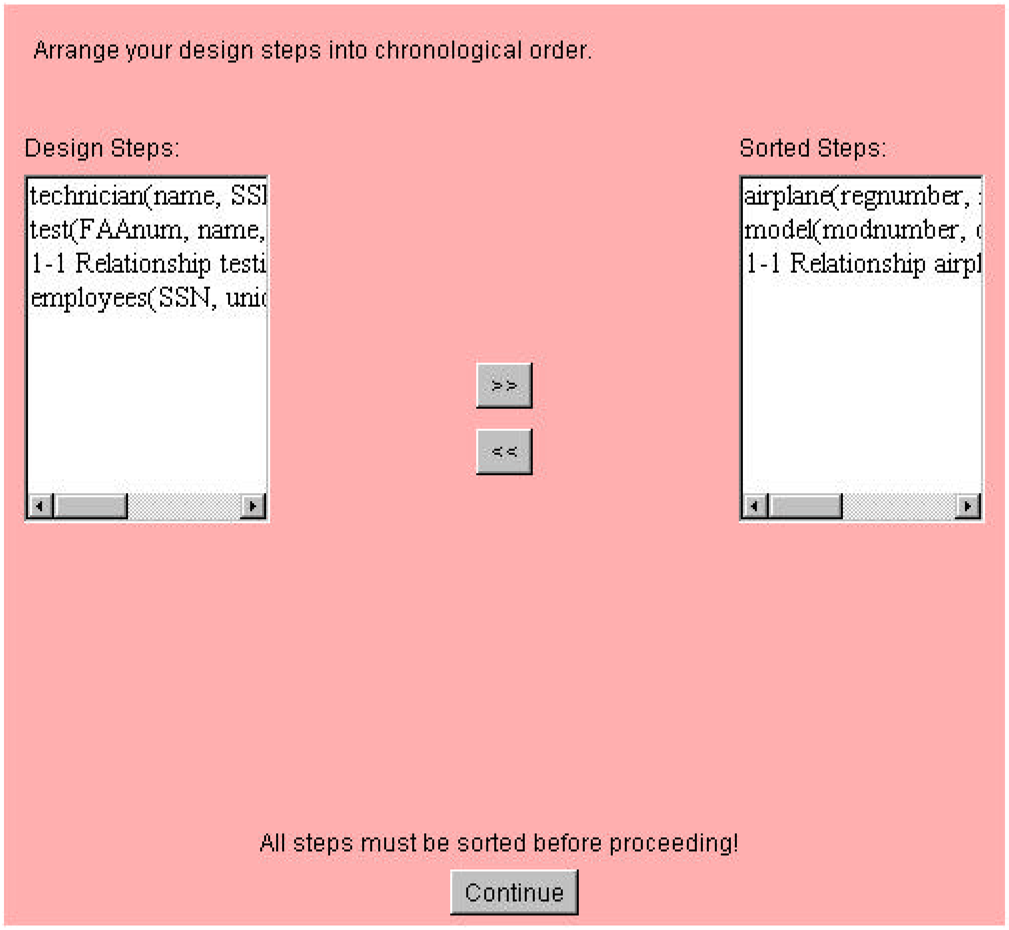
airplane(regnumber, r
model(modnumber, c
technician(name, SSN
1-1 Relationship airpl
test(FAAnum, name, r
1-1 Relationship testi

Figure 6-7. Step Entry Display

6.2.4.3 Step Ordering Display

The user may not have entered their steps in the order in which they would perform them. The Step Ordering Display asks them to take their list of steps and arrange them in the order in which they would normally be performed. This is done by moving them from a list displaying them in their original order into a second list. They can be moved in either direction so that if a step is moved into the wrong position it can be removed and added again. All steps are placed at the end of the list when moved over.

Figure 6-8 shows the Step Ordering Display.



Arrange your design steps into chronological order.

Design Steps:

technician(name, SSI
test(FAAnum, name,
1-1 Relationship testi
employees(SSN, unio

Sorted Steps:

airplane(regnumber, :
model(modnumber, c
1-1 Relationship airpl

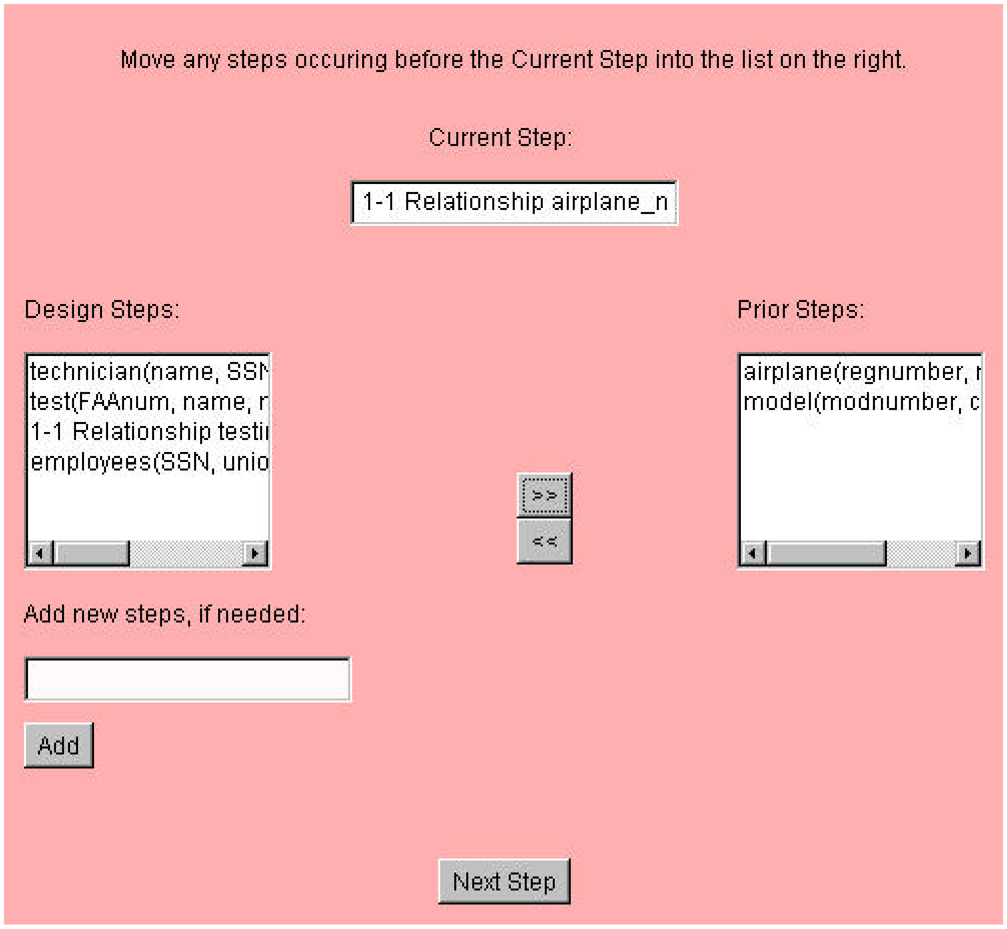
All steps must be sorted before proceeding!

Figure 6-8. Step Ordering Display

6.2.4.4 Card Sort Display

The Card Sort Display presents the subject with each step (the "current step") one at a time and asks them to indicate which steps must occur prior to the current one. This is done by moving the steps from a list containing all steps to a new list of prior steps. If the subject realizes that a step is missing, they are allowed to add it.

Figure 6-9 shows the Card Sort Display.


Move any steps occuring before the Current Step into the list on the right.

Current Step:

1-1 Relationship airplane_n

Design Steps:

technician(name, SSN
test(FAAnum, name, r
1-1 Relationship testi
employees(SSN, unio

Prior Steps:

airplane(regnumber, r
model(modnumber, c

Add new steps, if needed:

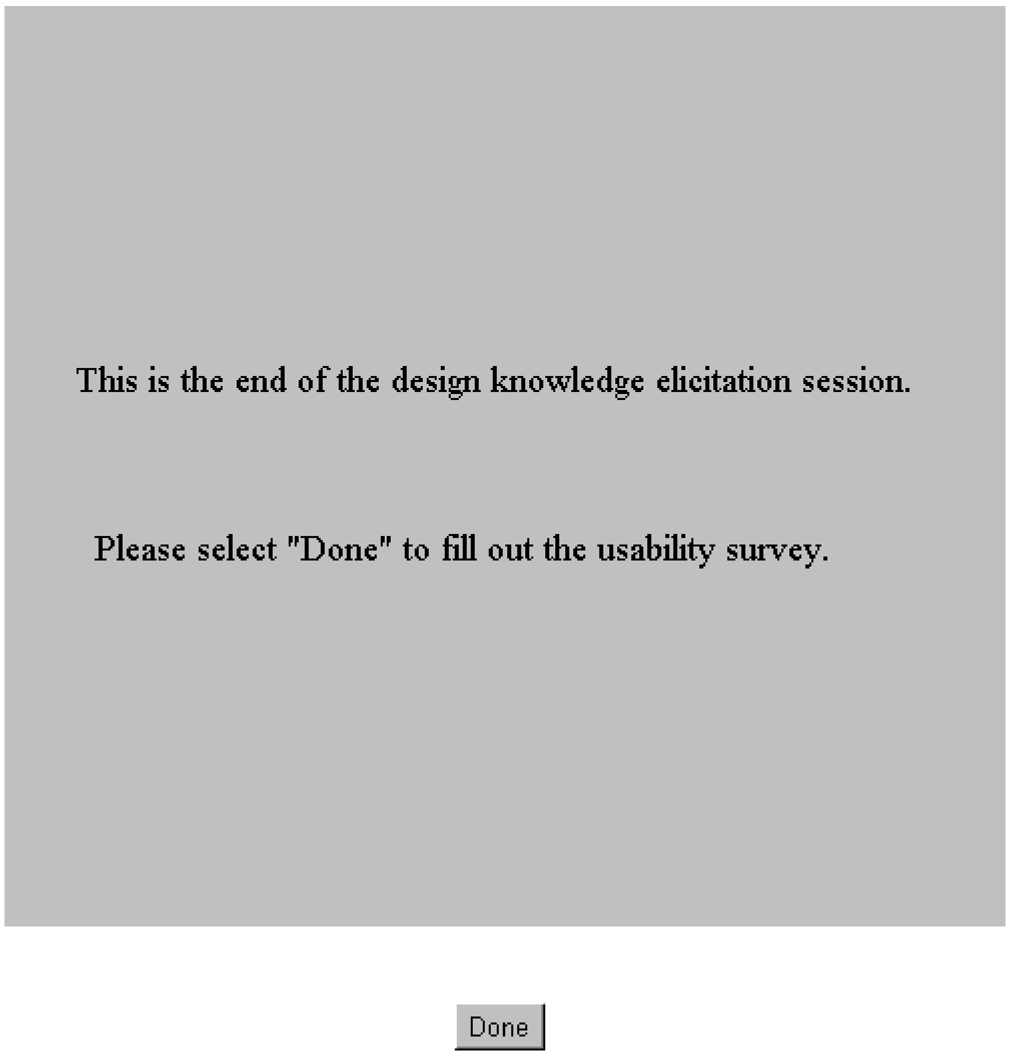
Figure 6-9. Card Sort Display

If steps are added, the sorting process needs to be repeated to incorporate the new step. This continues until all steps have been sorted without any new steps being added.

At this point, each step, and the set of steps considered to be prior to it, are written to the knowledge file.

6.2.4.5 Experiment Complete Display

After the steps have been sorted and written to a file, a final Java display is shown that instructs the subject to proceed to the usability survey. Figure 6-10 shows the Experiment Complete Display.


This is the end of the design knowledge elicitation session.

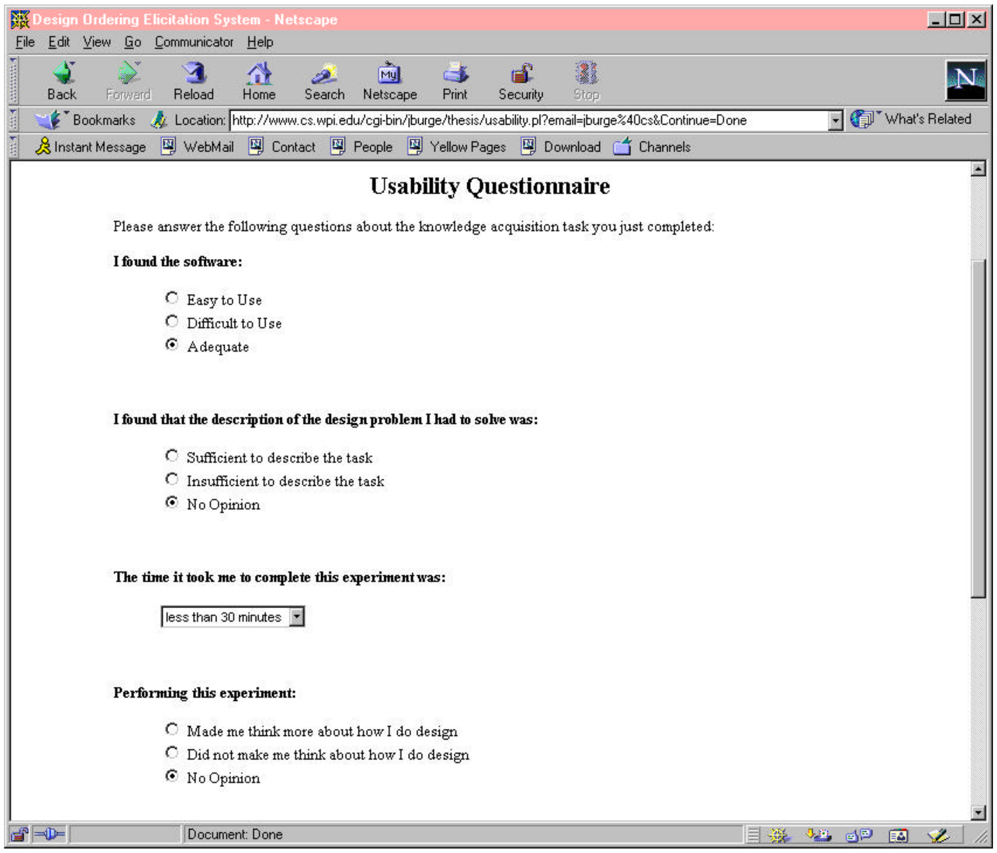
Please select "Done" to fill out the usability survey.

Figure 6-10. Experiment Complete Display

6.2.5 Usability Survey Display

The usability survey was written using a Perl script that generates the HTML form for the survey. All questions, except for the optional comments, are done using radio buttons and pull-down menus so JavaScript error checking is not needed. All survey results are written to an output file. The name of the output file is created by appending "_survey.dat" to the user ID entered during the background survey.

Figures 6-11 and 6-12 show the Usability Survey Display.


Design Ordering Elicitation System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://www.cs.wpi.edu/cgi-bin/jburge/thesis/usability.pl?email=jburge%40cs&Continue=Done
What's Related
Instant Message WebMail Contact People Yellow Pages Download Channels
Document: Done

Usability Questionnaire

Please answer the following questions about the knowledge acquisition task you just completed:

I found the software:

- Easy to Use
- Difficult to Use
- [✓ Adequate]

I found that the description of the design problem I had to solve was:

- Sufficient to describe the task
- Insufficient to describe the task
- [✓ No Opinion]

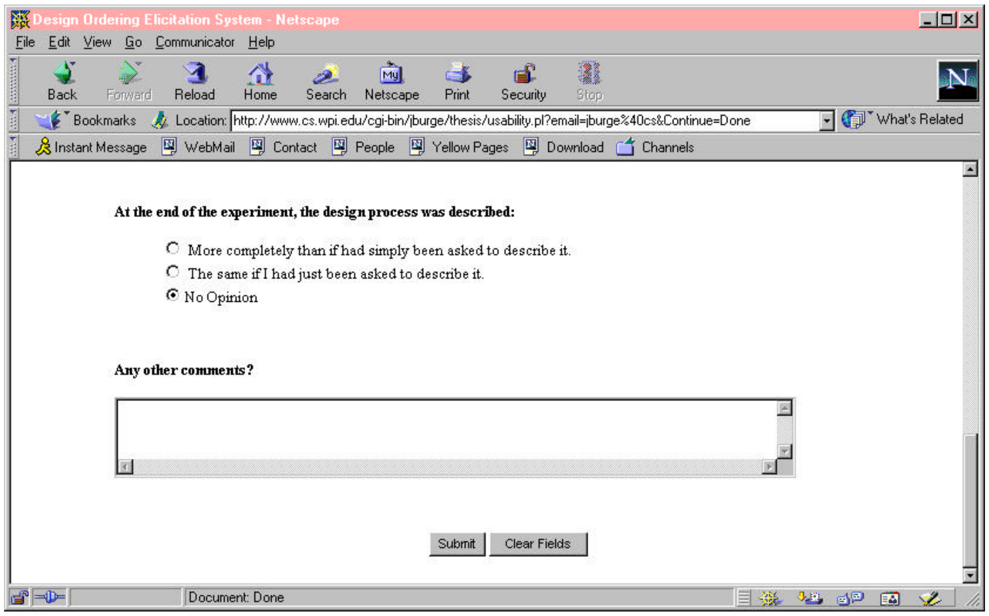
The time it took me to complete this experiment was:

less than 30 minutes

Performing this experiment:

- Made me think more about how I do design
- Did not make me think about how I do design
- [✓ No Opinion]

Figure 6-11. Usability Survey Display, Page 1


Design Ordering Elicitation System - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://www.cs.wpi.edu/cgi-bin/jburge/thesis/usability.pl?email=jburge%40cs&Continue=Done What's Related
Instant Message WebMail Contact People Yellow Pages Download Channels
Submit Clear Fields
Document: Done

At the end of the experiment, the design process was described:

- More completely than if had simply been asked to describe it.
- The same if I had just been asked to describe it.
- [✓ No Opinion]

Any other comments?

Figure 6-12. Usability Survey Display, Page 2

6.3 Implementation Issues

Java is an easy language to learn and use, especially when compared to other languages and toolkits such as Visual C++ and X-Windows. There were still several difficulties encountered while implementing DOES. The major ones were Java security issues and version incompatibilities.

6.3.1 Java Security

Since Java is run on the remote client machine, there are built-in security features that exist to protect both the client and the server. The problem that arose during this implementation was the restriction on file I/O. There was no way to write the experiment results directly to a file from Java. This was worked around by writing Perl scripts to perform file I/O. The scripts were invoked via HTTP. This worked well for small amounts of data but did not work for larger sets. In particular, the data transfer did not work when the final set of steps and prior steps was sent to the server. This was due to limitations on the amount of data that could be sent in a single HTTP request. The solution to this problem was to send each step and its associated prior steps to the server individually. This meant that there was a HTTP request sent for each step. This was time consuming and did not always work on remote machines.

6.3.2 Version Incompatibilities

Another difficulty was in Java version incompatibilities. Most Java books and classes teach the Java 1.1 event model. The Java code for DOES was written using the Java 1.0 event model. This was done because the browser version installed on several of the WPI Computer Science department PCs did not support Java 1.1. Asking research subjects to install a new web browser before running the experiment was not considered a reasonable request.

This had two impacts on DOES development: first, any code based on manual examples or downloaded from the web had to be modified to use the 1.0 event model. Second, this limited what features could be used for DOES. In particular, cut and paste was not available in 1.0. If the 1.1 event model had been used, the step sorting would have been done by allowing cut and paste directly into the correct spot in the list rather than moving steps from the original list to the sorted list.

Version incompatibilities continued to cause problems during DOES testing and deployment. In one case, DOES ran successfully on an older version of Netscape and did not run on a newer version. This problem was corrected but it is still not clear why the code was incompatible. There are also problems running DOES using the Internet Explorer browser. As a result, it is recommended that DOES be run using Netscape, version 4.0 or higher.

Chapter 7 DOES Experiment

This chapter discusses the experiment performed for this thesis.

7.1 Selecting the Subjects

Fourteen subjects were selected who had varying levels of experience performing database design. This was done so that the relationship between level of expertise and presence of implicit knowledge could be examined. All subjects were familiar with entity relationship design.

- Three Computer Science faculty members who specialize in databases;
- Three Computer Science Ph.D. students who specialize in databases;
- Five Computer Science graduate students who had taken or were in the process of taking an introductory database course;
- Three industry professionals who specialize in databases.

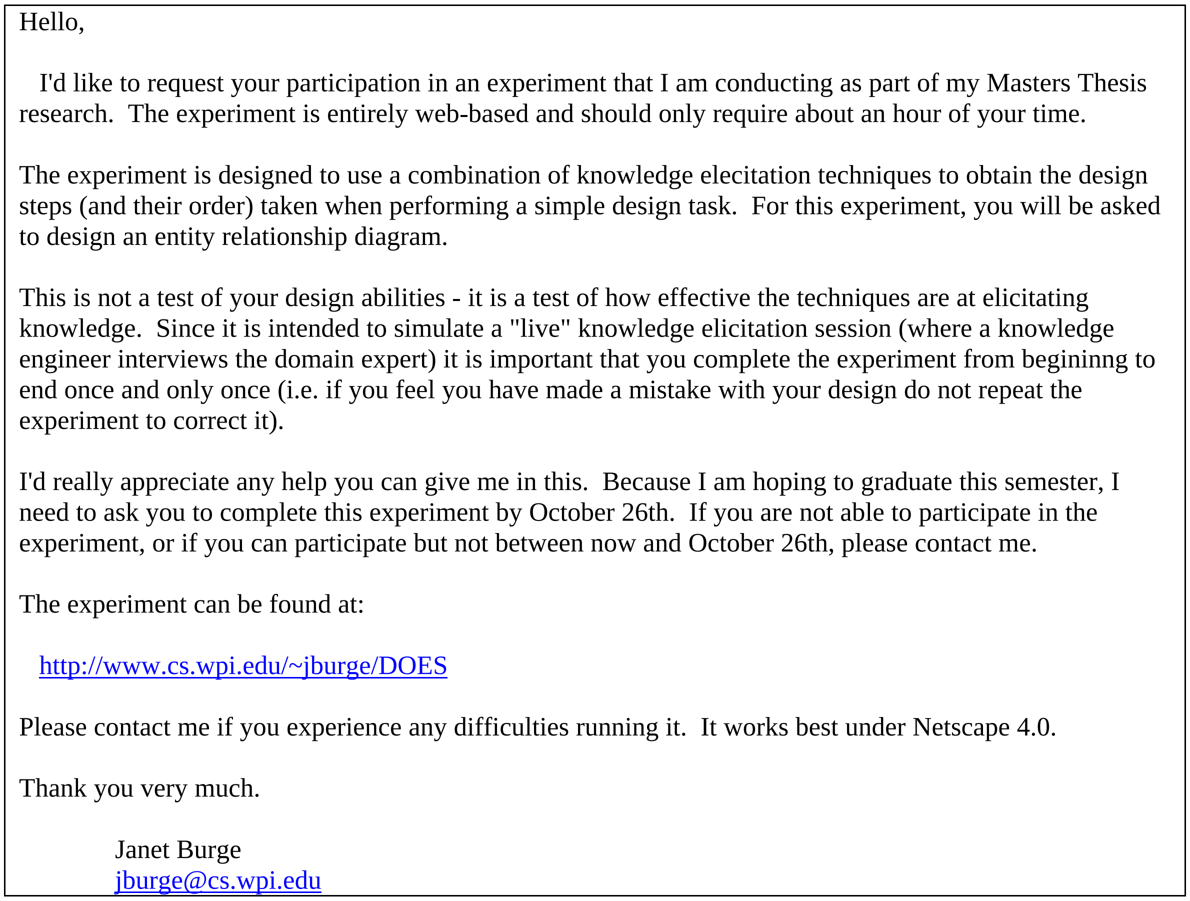
7.2 Stimuli Preparation

The design problem given to the subjects was a textbook exercise (number 14.8) from [Ramakrishnan, 1997, p. 392]. The exercise chosen was one that was sufficiently complex that different steps and orders of the steps could be produced by different subjects but not so complex that the exercise could not be completed in about one hour. The subjects were asked to create an entity relationship diagram for an airport database. The requirements for the database, a description of the specific task (i.e., create an entity relationship diagram), and examples of design steps were created and incorporated into the DOES experiment. Appendix A shows the task stimuli.

7.3 Experimental Procedure

Subjects were sent an e-mail message requesting their participation in the experiment. They accessed the experiment via the Web and did it at their convenience. They were requested to run the experiment only once unless told otherwise. All results were saved to files for analysis later. The experimenter was able to examine these files to determine who had and who had not taken the experiment and to determine if it had been completed correctly.

Figure 7-1 shows an example of the e-mail participation request.



Hello,

I'd like to request your participation in an experiment that I am conducting as part of my Masters Thesis research. The experiment is entirely web-based and should only require about an hour of your time.

The experiment is designed to use a combination of knowledge elecitation techniques to obtain the design steps (and their order) taken when performing a simple design task. For this experiment, you will be asked to design an entity relationship diagram.

This is not a test of your design abilities - it is a test of how effective the techniques are at elicitating knowledge. Since it is intended to simulate a "live" knowledge elicitation session (where a knowledge engineer interviews the domain expert) it is important that you complete the experiment from begininng to end once and only once (i.e. if you feel you have made a mistake with your design do not repeat the experiment to correct it).

I'd really appreciate any help you can give me in this. Because I am hoping to graduate this semester, I need to ask you to complete this experiment by October 26th. If you are not able to participate in the experiment, or if you can participate but not between now and October 26th, please contact me.

The experiment can be found at:

http://www.cs.wpi.edu/~jburge/DOES

Please contact me if you experience any difficulties running it. It works best under Netscape 4.0.

Thank you very much.

Janet Burge
jburge@cs.wpi.edu

Figure 7-1. Example Participation Request

7.4 Possible Outcomes

As with all experiments, there are many possible outcomes. For each subject, any one of the following outcomes might occur:

- Additional steps and additional orders are produced (that were not produced by the direct technique alone) - this would occur if implicit knowledge was involved in both the steps and the orders.

- Additional orders, but no additional steps produced - this would occur if implicit knowledge was involved in step ordering but was not present in the steps for this specific design task or if implicit knowledge was present but the experiment was not successful at obtaining it.
- No additional steps or additional orders were produced - this would occur if implicit knowledge was not involved for this design task or if the experiment was not successful at obtaining it.
- Results were produced but not usable - this would occur if there were problems with either the experiment or if the subject did not perform the experiment as instructed.
- The subject does not perform the experiment - some subjects may be unable to perform the experiment because of other obligations.

Chapter 8, experiment results, discusses the actual outcomes of the experiment and how they relate to the hypotheses proposed in Chapter 4.

Chapter 8 Results and Evaluation

This chapter discusses the results from each part of the experiment and discusses how they relate to the hypotheses proposed. Section 8.1 discusses the DOES subjects and the results of the background survey, section 8.2 discusses the KE experiment, and section 8.3 discusses the usability survey results. Figure 8-1 shows the DOES State Diagram and indicates which sections of this chapter discuss each part of the experiment.

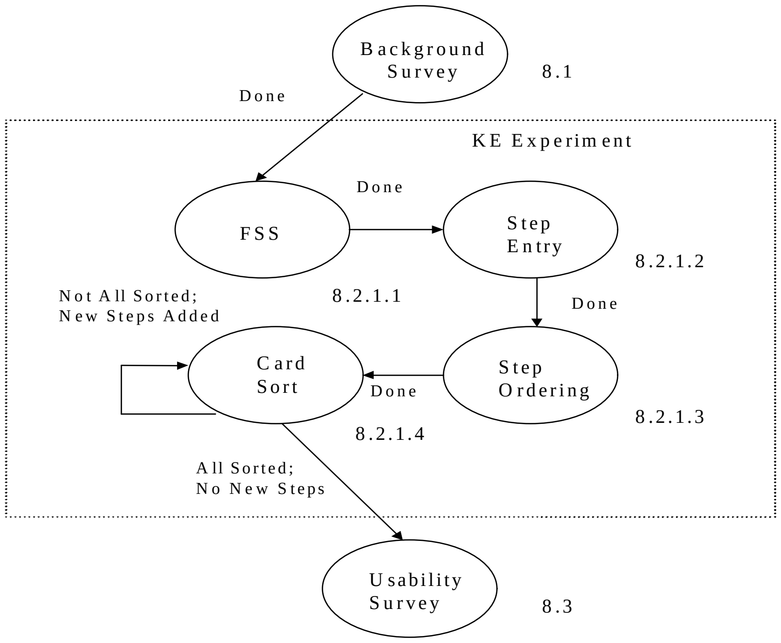


Figure 8-1. DOES State Diagram with Section Numbers

8.1 DOES Experiment Subjects

As described in the previous section, fourteen subjects with a variety of backgrounds were asked to participate in the experiment. Of these subjects, eleven subjects completed all or part of the experiment. Ten subjects completed the experiment remotely, i.e. without the presence of the experimenter. One of these ten had an office near the experimenter and walked over to ask questions during the experiment. The eleventh subject completed the experiment in the presence of the experimenter and was able to show the experimenter intermediate results and ask questions.

Table 8-1 shows the number of subjects completing each portion of the experiment.

Table 8-1. Number of Subjects Completing the Experiment

Experiment Portion	Number of Subjects
Background Survey	11
Forward Scenario Simulation	10
Step Entry	9
Step Ordering	7
Card Sort	5
Usability Survey	9

There were several reasons why subjects did not complete the experiment. Some subjects tried running the experiment using Internet Explorer rather than Netscape. Errors were not always reported, but the results were not recorded properly. Other subjects did not follow the instructions and complete the experiment: after entering the steps, they decided that since they had already entered them in order they did not need to order them again or perform the Card Sort portion of the experiment. Of the subjects who completed all portions of the experiment, two of them performed it twice: one because their solution was abstract, rather than the detailed solution requested, and one because they felt their result for the Forward Scenario Simulation had sufficient detail and therefore they did not need to complete the rest of the experiment. Only subjects who asked the experimenter if their results were complete were asked to repeat the experiment.

All eleven subjects completed a background survey as part of the experiment. Table 8-2 shows the education levels for the subjects who completed any portion of the experiment. The third column shows the education levels for subjects who completed all parts of the experiment.

Table 8-2. Education Level of Research Subjects

Level of Education	**Survey**	**Entire Experiment**
Ph.D.	3	1
MS.	6	3
BS.	2	1

Table 8-3 shows the number of years of relevant experience.

Table 8-3. Relevant Experience

Years of Experience	**Number of Subjects**
0	2
1-5	7
6-10	1
11-15	1

The default response to the question was zero, one zero response was for someone taking the introductory graduate database class, the other was for someone who had worked in databases and probably did not choose a response to the question. The majority of the respondents selected 1-5. This included two faculty members who specialized in databases. The subject who entered 11-15 is student currently taking the introductory database class. Their answer to the question on relevant training mentioned OOA and OOD classes so their answer may have included experience with related topics as well as with databases.

8.2 DOES KE Experiment Results

Results were obtained from both the direct technique as well as the direct and indirect technique combined. The following sections present the results for both and the comparison between them.

8.2.1 Results from Direct Technique

The direct technique consisted of three parts: Forward Scenario Simulation where the subjects were asked to describe how they would perform their design, Step Entry where subjects were asked to break their design into steps, and Step Ordering where they were asked to arrange their steps into chronological order.

8.2.1.1 Forward Scenario Simulation

For the Forward Scenario Simulation (FSS) part of the experiment, the subjects were asked to describe how they would design the entity relationship diagram for an airport database (described in Appendix A). Ten subjects entered a free-text description of their design process. The results can be classified into five categories:

- *detailed entity relationship diagram (ERD) descriptions* -- this was the result that was requested;
- *descriptions of the database design process* -- these descriptions included the mappings from the entity relationship diagrams to database tables and how the tables would be normalized;
- *abstract descriptions* -- high level descriptions of how an ERD would be constructed without referring to the specific problem that was given;
- *abstract descriptions of the object oriented design process* -- object diagrams have similar features but this is not the problem that was proposed;
- *incomplete descriptions* -- not all of the needed information was present.

Table 8-4 summarizes the results.

Table 8-4. Forward Scenario Simulation Results

Category	**Number of Subjects**
Detailed ERD Description	1
Database Design Description	4
Abstract ERD Description	2
Abstract Object Oriented Design Description	1
Incomplete Description	2

This portion of the experiment was intended as a warm-up exercise so the subjects were not required to enter detailed descriptions. It is clear from the results, however, that several subjects did not understand that the task was to design an ER diagram, not the entire database.

8.2.1.2 Initial Step Entry

After entering their initial description of the design task, the subjects were asked to break their design down into discrete steps. Nine subjects completed this portion of the experiment. Even though all subjects were given instructions that the task was to design an entity relationship diagram and were given examples of the level of detail needed for design steps, the solutions were not consistent in the task performed or the level of detail. These solutions can be broken into three categories: abstract steps (descriptions of the ERD process that did not refer to the problem given), detailed ERD designs, and detailed ERD designs with foreign keys (relationships between database tables). Table 8-5 summarizes the results.

Table 8-5. Initial Step Entry Results

Category	**Number of Subjects**	**Number of Steps**
Abstract Steps	2	6, 8
Detailed ERD Designs	4	11 (2), 14, 43
Detailed ERD Designs with Foreign Keys	3	16, 19, 25

The number of steps given was consistent for each category with the notable exception of the subject who reported 43 steps for his design. This subject started by listing each attribute definition as a separate step, rather than listing them along with the entity as demonstrated by the example steps given. For example, rather than specifying the entity airport as airport(ID, name, address, phone) they listed airport entity, airport DI, airport name, airport address, airport phone all as separate steps. This accounted for 22 of the 43 steps. He also specified "looking into requirements" as a step and specified an abstract step of identifying relations. If these steps are subtracted, 19 steps remain, which is closer to the results from the other subjects and more consistent with the instructions.

8.2.1.3 Step Sorting

After the subjects entered their steps, they were then asked to arrange them into the order in which they would normally be performed. Seven subjects completed this part of the experiment. Table 8-6 summarizes the results.

Table 8-6. Step Sorting Results

Category	Number of Subjects
No change in order	2
Order changed to correct original order	1
Order changed, reason unknown	1
Relationships and/or keys moved to directly follow the entities involved	3

All but one subject who completed this portion of the experiment had approached the problem at a detailed level. The subject who provided abstract steps did not change their order.

8.2.1.4 Evaluation of Solutions

Solutions were evaluated against the solution provided by [Ramakrishnan, 1987], shown in Figure 8-2.

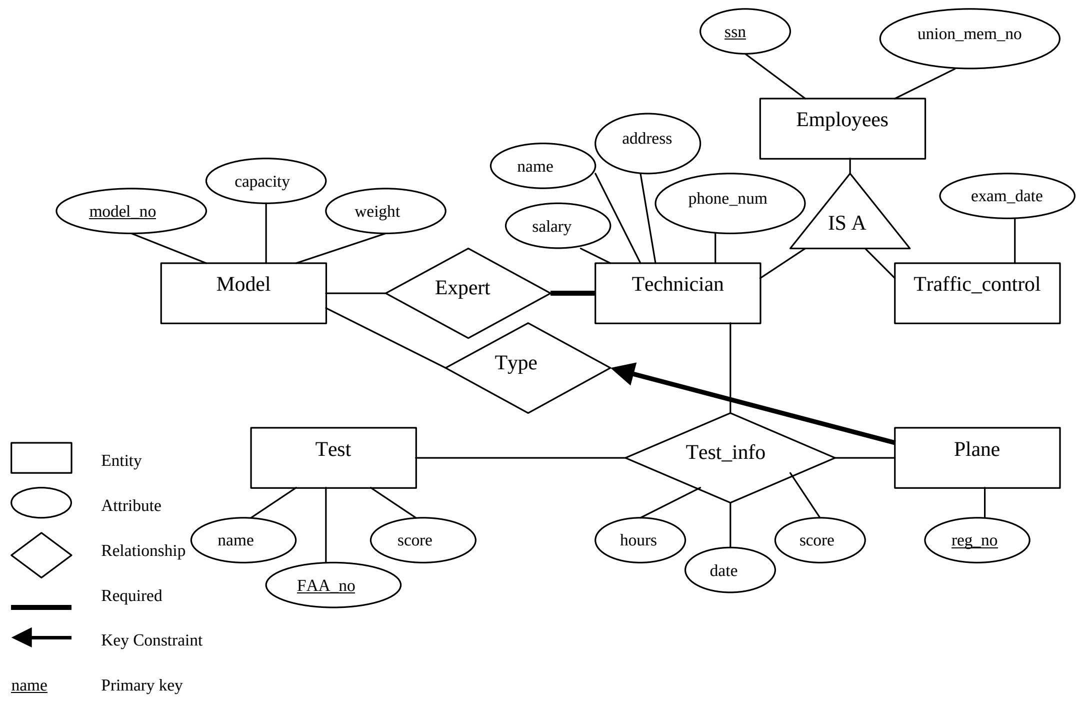


Figure 8-2. ER Diagram for Airport Database

The results were as follows:

- One subject gave an answer that matched the book solution exactly
- One subject missed the relationship specifying who the plane was tested by.
- One subject missed the relationship between test and plane.
- One subject missed the Test entity completely. This subject also had many items that could have been attributes defined as entities.
- Three subjects had difficulties with the relationships between employees, technicians, and traffic controllers. Two of these solutions still met the requirements, they were just less efficient.

Of these subjects, three gave solutions that would meet all the requirements. Most gave solutions that were close. Two of the subjects giving correct solutions were Ph.D. students specializing in databases. The student who gave the exact solution was a student who was currently taking the introductory graduate database course.

8.2.2 Results from Indirect Technique

Five of the eleven subjects completed the Card Sort portion of the experiment. In this part of the experiment, each step was displayed one by one to the subject. They were asked to choose which steps from the remaining steps had to occur prior to the presented step.

The results were analyzed to determine the following: were any additional step added and were any additional orders obtained? No additional steps were entered by any of the subjects completing this part of the experiment. Four out of the five subjects chose what they believed to be the minimal set of steps that had to occur before each step. For these subjects, many different orders could be derived for the design task. The fifth subject did not select any of the steps as being prior to any other steps.

Of the four subjects who provided dependencies, one had an obvious error in the order specified in the Card Sort: this subject indicated that a primary key should be created without specifying that the corresponding relation must be created first. Another subject chose prior steps that conflicted with the order specified when they sorted the steps originally. In these results, the Card Sort order dependencies were correct. The third subject chose an incorrect entity as being required before a relationship was defined.

It appears that they simply moved the wrong item over to the list of prior steps by mistake (there were no other errors). The fourth subject (who did not perform the experiment remotely) made one error by not moving a step over to the prior step list when they should have. This same subject (who had 43 steps total) got tired of indicating all the prior steps and started to only indicate the most recent prior step; this was observed by the person administering the experiment.

Figure 8-3 shows an example of a provided order and figure 8-4 shows an example of an order that was derived from the Card Sort results.

1. Create an entity Model with attributes (ModelNumber, capacity, weight)
2. create primary key Model.ModelNumber
3. create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
4. create primary key Airplane.RegistrationNumber
5. create an m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
6. Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)
7. create primary key Technician.SSN
8. create an m-to-n relationship TechExpert bewteen Technician and Model
9. Create an entity TechExpert with attributes (SSN, ModelNumber)
10. create an n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
11. create an n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
12. Create an entity TrafficController with attributes (SSN, DateOfExam)
13. create an entity Employees with attributes (SSN, unionMembershipID)
14. create an entity Test with attributes (FAANumber, Name, MaxScore)
15. create primary key Test.FAANumber
16. create an m-to-m relationship TestingEvent between Test and Airplane
17. create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
18. create an n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
19. cretae an n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
20. create an n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber

Figure 8-3. Sorted Steps from Direct Method

1. create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
2. create primary key Airplane.RegistrationNumber
3. create an entity Test with attributes (FAANumber, Name, MaxScore)
4. create an entity Employees with attributes (SSN, unionMembershipID)
5. create primary key Test.FAANumber
6. Create an entity TrafficController with attributes (SSN, DateOfExam)
7. Create an entity Model with attributes (ModelNumber, capacity, weight)
8. create primary key Model.ModelNumber
9. create an m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
10. Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)
11. create primary key Technician.SSN
12. create an m-to-n relationship TechExpert bewteen Technician and Model
13. Create an entity TechExpert with attributes (SSN, ModelNumber)
14. create an n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
15. create an n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
16. create an m-to-m relationship TestingEvent between Test and Airplane
17. create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
18. create an n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
19. cretae an n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
20. create an n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber

Figure 8-4. Derived Order from Card Sort Results

8.2.3 Comparison between Methods

By isolating the dependencies, the indirect method produced the information needed to create multiple step orderings for the task. Unfortunately, the technique was also prone to errors and inconsistencies. In one case, there were two discrepancies between the two techniques. The differences in order are shown in Table 8-7.

Table 8-7. Direct/Indirect Discrepancies

Steps and Order from Direct	Steps and Order from Indirect
• Create an Airplane entity with attributes registration # and model • Create a Model entity with attributes model #, capacity, and weight; model # is the key	• Create a Model entity with attributes model #, capacity, and weight; model # is the key • Create an Airplane entity with attributes registration # and model
• Create a Technician entity with attributes name, SSN, address, phone number, salary • Think about the address attribute. It might be just a simple attribute or a separate entity, if each technician might have more than one address	• Think about the address attribute. It might be just a simple attribute or a separate entity, if each technician might have more than one address • Create a Technician entity with attributes name, SSN, address, phone number, salary

The difficulty with the steps in the first row of the table is the way the problem statement was ordered. The problem statement indicated that an airplane exists and it has a registration number and is of a specific model. The problem statement then indicated that each model has a model ID, capacity, and weight. The resulting steps provide be the correct way to design the database tables, and either of the two orders would be correct, but for the entity relationship diagram, the airplane should not have had a model number attribute. Instead, there should have been a relationship between airplane and model.

For the second set of steps, they could have been done in any order. It would be possible to create the technician entity and then think about the address format or to think about the format and then decide to make it an attribute.

The subject was asked why there was a discrepancy between the two methods. The reason given was that the sorted steps given in the direct method were the way that he had performed the design, the dependencies given in the indirect method were the way the design should have been done. This was the only subject who gave a detailed solution who did not change the order of their steps when given the opportunity to do so after initially entering them.

In summary, using two techniques increased the number of orders produced for the design steps. It also produced errors and inconsistencies that could result in incorrect orders. A way to avoid or correct these errors needs to be provided.

8.2.4 Relationship Between Results and Hypotheses

The following hypothesis were proposed in Chapter 4:

- Design plan steps and their order involve implicit knowledge.
- The combination of a direct technique with an indirect technique obtains more information than using the direct technique alone.
- The level of expertise affects the presence of implicit knowledge.
- The way KE is done (remote vs. in person) affects the quality of the results obtained.
- The amount of implicit knowledge involved depends on the type of domain
- The amount of implicit knowledge involved depends on the type of task

The following sections discuss the first four hypothesis and how they relate to the results of the experiment. The remaining two are discussed in Chapter 9 as potential future work.

8.2.4.1 Implicit Knowledge in Design Steps and Ordering

The indirect approach did not add any additional steps to this design. This could be for several reasons. The subjects may have been sure enough of their results that they did not look for the possibility of missing information. They also may not have been "expert enough" to perform the task automatically. The type of problem chosen may not have been one where implicit knowledge is involved in determining the steps. Another possibility is that implicit knowledge is involved but is not seen as something that needs to be stated when specifying the steps to the problem.

The two-method combination was effective at determining the dependencies needed to derive multiple orders of design steps. These alternative orders can be considered t_2-implicit with respect to this experiment since two KE techniques were applied without directly obtaining the information.

8.2.4.2 More Information by Combining Techniques

Adding the indirect technique did not result in any additional steps. It did, however, provide the information needed to generate many alternative orders. This indicates that adding the second technique did obtain more information. This did come at a cost -- the indirect technique meant that the subjects had to specify the dependencies for each step. For problems with many steps this could be difficult. It was also prone to error. This may have been because the subjects were all volunteers and were in a hurry to complete the experiment and get back to their other work. One solution to this problem would have been to ask them to verify all results - this would result in fewer errors but would increase the time needed to complete the experiment.

8.2.4.3 Impact of Level of Expertise

The four subjects who completed the implicit portion of the experiment were one Ph.D. specializing in databases, two Ph.D. students specializing in databases, and one Ph.D. student who had studied databases but did not specialize in them. All subjects chose, or tried to choose, the minimal number of prior steps for each design step.

The level of expertise did appear to have an impact on the number of design steps provided but did not affect the quality of the solution. The "best" solution was provided in a small number of steps by a MS student who was just taking the database class. One reason for this may have been that the student had recently learned ERD design from the same book that the exercise was from.

8.2.4.4 Evaluation of Remote KE

The remote KE approach proved to have many problems. The first difficulty was in getting subjects to participate. All subjects contacted to perform the experiment either did so or responded via e-mail that they would participate. Despite this, only eleven out of fourteen started the experiment and only five produced results for all parts. Of the seven subjects who produced detailed solutions, only three met all the stated requirements. Of the four who used the Card Sort to sort their steps, two had errors. A third had inconsistencies but was able to explain them. The fourth had assistance from the knowledge engineer during the experiment.

Even though the subjects were presented with detailed instructions, many of the subjects did not follow them. Four subjects did not follow the instructions, which stated the level of abstraction to use in their solution. Two of these subjects were asked to re-take the experiment.

Data was lost for several subjects. Two of them were performing the experiment from an off-campus location, which may have caused data transfer problems. Another performed the experiment using Internet Explorer and experienced Java errors.

Several subjects did not complete the project because they decided some parts were not necessary. Two entered their steps into the free-text area and then skipped the rest of the experiment because they had already provided steps. One entered the discrete steps but then stopped when asked to sort them because they had already entered them in the order in which they should be performed. One entered the discrete steps and sorted them, but did not provide any dependencies.

Some of these problems may have been avoided if the KE sessions had been scheduled. The subjects could have been encouraged to spend more time reading the instructions. They also could have been prompted to provide detailed solutions and complete the experiment.

Because of these difficulties, one subject performed the experiment in the presence of the knowledge engineer. The following difficulties and misunderstandings were observed:

- The subject did not refer to the experiment stimuli (requirements for the airport database, description of the design task, and example steps) at all during the experiment. He had looked through it earlier and relied completely on his memory.
- The subject did not like having to type in their steps and started combining steps in order to save typing.
- During the Card Sort exercise, the subject started to just give back the same order that he gave earlier. He had to be instructed to only state a step as being prior to another step if there were dependencies involved.
- During the Card Sort, the subject listed all the dependencies for the first few steps but started only specifying the closest prior step to save time.

Performing the experiment under the supervision of the knowledge engineer resulted in a complete experiment at the correct level of detail. It also resulted in a Card Sort that correctly identified dependencies (although, for some, only the nearest dependency in the chain). It did not, however, result in a correct solution. This was probably because the subject relied on his memory and did not want to consult the instructions (although prompted to) during the experiment. He may have wanted to show that he did not need to look at them to complete the experiment.

8.3 Usability Survey Results

The usability survey had six questions. Nine subjects completed the survey. The following sections give the results for each question.

8.3.1 Overall Usability

The first question asked for an overall assessment of the usability of the system. Table 8-8 summarizes the results.

Table 8-8. Usability Results

Response	**Number of Subjects**
The system was easy to use.	2
The system was neutral to use.	3
The system was hard to use.	4

These results are not surprising considering the difficulty that many of the subjects had in completing the experiment. A major factor in this was the narrowness of some of the input fields. Java does not provide a way to specify a field width when creating lists and no other way to do it was found when the experiment was created.

8.3.2 Design Task Description Evaluation

The second question asked for an assessment of the data provided that described the requirements and task specification for the design problem. Table 8-9 summarizes the results.

Table 8-9. Description Evaluation Results

Response	Number of Subjects
Data provided was sufficient to solve the problem	7
Neutral response	0
Data provided was not sufficient to solve the problem	2

From these responses, it seems that most subjects found the data provided to be adequate. Of the two who did not, one commented that she had needed to make some assumptions in order to complete her solution. The other person only provided high level steps and appeared to be performing an object oriented design project, not a database design project.

8.3.3 Amount of Time Needed to Complete the Experiment

The experiment was designed so it could be completed in one hour. In order to verify that estimate, the third question asked the subjects to give the amount of time needed to complete the project. Table 8-10 summarizes the results.

Table 8-10. Amount of Time Needed

Results	Number of Subjects
Less than 30 minutes	2
31-60 minutes	4
61-90 minutes	3

The time-stamps from the data files were examined to obtain the actual time used to complete the experiment. Three of the subjects underestimated the time that they used by a small amount. One subject indicated that they used 61-90 minutes but actually used only 19. They may have experienced difficulties using the software and had to repeat the experiment.

8.3.4 Prompted Thought about Design Process

The fourth question in the survey asked the subjects if performing the experiment made them think more about how they did design. Table 8-11 summarizes the results.

Table 8-11. Prompted Thought About Design Process

Response	**Number of Subjects**
Yes	6
No-opinion	1
No	2

Most subjects responded that performing the experiment made them think more about how they performed design.

8.3.5 Completeness of Description

The fifth question asked the subjects if, at the end of the experiment, the design process was described more completely than if they had simply been asked about it, the same as if they had just been asked about it, or if they had no opinion. Table 8-12 summarizes the results.

Table 8-12. Completeness of Description

Response	**Number of Subjects**
Better description	3
No opinion	3
Same description	3

Of the four subjects who completed the Card Sort portion of the experiment, two felt that the experiment provided a better description, had no opinion, and the other thought it was the same.

8.3.6 Comments

Three subjects did not provide any comments on the experiment. Those who did had the following comments:

- Text entry fields needed to be wider (3 subjects)
- General questions/comments on their results (2 subjects)
- Explanation of assumptions about the task that were made (1 subject)
- Text of a Java error message encountered when using Internet Explorer

In the next chapter, ideas for future work on DOES are discussed.

Chapter 9 Future Research

This chapter presents some ideas for future work on DOES. These include DOES enhancements, automated analysis of results, and additional experiments.

9.1 DOES Enhancements

The research subjects had some problems with understanding how to use DOES and with using the interface. The following list suggests some improvements to the user interface, the result collection, and the documentation of the experiment.

- Increase the width of data entry and display fields - there does not seem to be an easy way to do this in Java but it must be possible.
- Test DOES on additional platforms to avoid difficulties like those experienced by subjects using Internet Explorer.
- Take advantage of Java 1.1 features - this would make the application easier to use.
- Add a log-in to the project to prevent subjects from taking the experiment more than once - this has to be balanced with the inconvenience to the subjects that this might cause.
- Record the time at the start and end of each part of the experiment - this can be used to determine where most of the time is spent.
- Come up with a more robust method for saving data to text files.

- Write the usability survey in Java so the user will be forced to complete all the proceeding sections of the experiment before proceeding to the survey.

9.2 Automated Analysis

The results from DOES consist of the dependency information needed to obtain all the alternative orders for the design steps. Determining these orders is not easily done manually - depending on the number of steps and dependencies the number of orders could be quite large. If the goal is to enumerate all the orders, software needs to be written to automate this process.

9.3 Additional Experiments

There are several additional experiments that could be performed using DOES:

- *Additional subjects*: the number of subjects completing this experiment was small. The experiment should be run (successfully) with a larger number of subjects so that results obtained are representative of the general population of domain experts.
- *Different domains*: the amount of implicit knowledge involved in design varies depending on the domain. For this experiment, the domain chosen was software, more specifically databases. It would be interesting to try using DOES in other domains to see how the results vary.
- *Different tasks*: the amount of implicit knowledge involved might also vary depending on the design task. The ERD task used here was one that was

fairly straightforward to solve. If the subjects had been asked to perform a different task, perhaps database tuning or physical database design, the results may have been different.

- *Different KE techniques/combinations of KE techniques*: substituting a different technique for the FSS or Card Sort would probably produce different results. The techniques could be compared to determine the most effective combination.

The next chapter presents the conclusions of this thesis.

Chapter 10 Conclusions

This chapter examines the results from the experiment in two major areas: the effectiveness at obtaining the design steps and ordering and in the effectiveness of the remote KE approach.

10.1 Design Steps and Ordering

The results from the direct method (Forward Scenario Simulation) and the direct method with the indirect method (Card Sort) were compared. Using the indirect method did not result in discovering any missing design steps. It did, however, result in producing the data needed to produce many different orders for the design steps. These additional orders can be considered t_2-implicit knowledge. While many subjects had a difficult time completing the experiment, six out of nine felt that it made them think about how they performed design.

10.2 The Remote KE Approach

Performing the experiment remotely was less than successful. Out of the ten subjects who started the KE portion of the experiment, only five finished it completely and of the five, one stepped through the last portion without entering any information. Of the four who produced useful results, one performed the experiment under the supervision of the knowledge engineer. This seems to indicate that despite its convenience, remote KE is not necessarily the most effective approach. The experiment could be modified to provide more guidance to the user; the drawback to this is that if they are given too much information on the goals of the experiment it may bias their results.

Part of the reason for the difficulty in performing the experiment is that all subjects were volunteers and had many tasks that were a higher priority than completing this experiment. If time had been scheduled for each subject to perform the experiment, they may have been more likely to spend adequate time on it. This may have resulted in fewer user errors and more time spent reading the instructions.

10.3 Summary

In conclusion, DOES, and the combination of KE methods it used, was successful at obtaining multiple orders of design steps. It was not as successful as a remote KE tool. There were many problems encountered by the subjects during the experiment. These ranged from browser crashes to misinterpreted instructions to lost data. Doing the application in Java allowed more control of the experiment than if HTML forms had been used but also caused problems with data transfer, input field size, and browser incompatibilities.

The DOES results, although fewer than desired, were good. Although no new steps were produced during the Card Sort, this may have been do to the fact that the task was not difficult or that the research subjects did not have a lot of time to spend on the project. All of the subjects who provided the dependencies for their design steps produced the minimal set of dependencies. This shows that the two KE techniques chosen, Forward Scenario Simulation and Card Sort were successful in producing the data required to obtain the maximum number of possible orders for the design steps. It is clear from our experiment that combinations of direct and indirect methods are useful but more work needs to be done to determine productive combinations.

Chapter 11 References

Anderson, J. (1983). *The architecture of cognition.* Cambridge, MA: Harvard University Press.

Atkinson, G. (1990). Practical experience using an automated knowledge acquisition tool, *Proceedings of the Second Annual Conference of the International Association of Knowledge Engineers*, pp. 87-97.

Bainbridge, L. (1979). Verbal reports as evidence of the process operator's knowledge, *International Journal of-Man-Machine Studies*, 11, pp. 411-436.

Belkin, N. J., Brooks, H. M. (1988). Knowledge elicitation using discourse analysis, In B. Gaines and J. Boose (Eds.) *Knowledge Based Systems,* Vol. 1, pp. 107-124. Academic Press Limited.

Bernaras, A. (1993). Models of Design for the CommonKADS Library, ESPIRIT Project P5248 KADS-II.

Berry, D. (1987). The problem of implicit knowledge, *Expert Systems,* August 1987, Vol. 4, No. 3.

Blessing, L. (1996). Design Process Capture and Support, *Proceedings of the 2nd Workshop on Product Structuring,* Delft, June 1996, M. Tichem et. al (Eds), pp. 109-121.

Blessing, L (1994). A Process-Based Approach to Computer-Supported Engineering Design, Cambridge: Black Bear Press.

Boose, J.H. (1989). A survey of knowledge acquisition techniques and tools, In Buchanan, B., Wilkins, D. (Ed.), *Readings in Knowledge Acquisition and Learning,* California: Morgan Kaufmann, pp. 39-56.

Breuker, J., Weilinga, B. (1983). Analysis techniques for knowledge based systems. Part 2: Methods for knowledge acquisition, Esprit Project 12, Report 1.2, University of Amsterdam.

Brown, D., Chandrasekaran, B. (1989). *Design Problem Solving: Knowledge Structures and Control Strategies*, California: Morgan Kaufmann.

Brown, D. (1992). Design, In Shapiro, S. (Ed.), *Encyclopedia of Artificial*

Intelligence, Vol. 1, New York: John Wiley & Sons, pp. 331-339.

Brown, D. (1993). Intelligent Computer Aided Design, *Encyclopedia of Computer Science and Technology*, Marcel Dekker, Inc., pp. 153-166.

Brown, D. (1998). Personal Communication, October, 1998.

Burton, A.M., Shadbolt, N. R. (1987). Knowledge Engineering, In N. Williams and P. Holt (Eds.), *Expert systems for users*, London: McGraw Hill.

Chandrasekaran, B. (1990) Design Problem Solving: A Task Analysis, *AI Magazine*, pp. 59-71.

Chen, A., McGinnis, B., Ullman, D., Dieterich, T. (1990). Design History Knowledge Representation and Its Basic Computer Implementation, *The 2nd International Conference on Design Theory and Methodology*, ASME, Chicago, IL, pp. 175-185.

Chignell, M. H., Peterson, J. G. (1988). Strategic issues in knowledge engineering, *Human Factors*, 30(4), 381-394.

Coovert, M. D., Cannon-Bowers, J. A., & Salas, E. (1990). Applying mathematical modeling technology to the study of team training and performance, Paper presented at *The 12th Annual Interservice/Industry Training Systems Conference*, Orlando, FL, November.

Cordingley, E. S. (1989). Knowledge elicitation techniques for knowledge-based systems, In D. Diaper (Ed.), *Knowledge elicitation: Principles, techniques and applications*, Chichester, England: Ellis Horwood Ltd.

Coury, B. G., Motte, S., & Seiford, L. M. (1991). Capturing and representing decision processes in the design of an information system, *In Proceedings of the Human Factors Society 35th Annual Meeting*, Santa Monica, CA: Human Factors Society, pp. 1223-1227.

Cross, N., Christiaans, H. , Dorst, K. (1996). Introduction: The Delft Protocols Workshop, In Cross, N., Christiaans, H. , Dorst, K. (Ed.), *Analysing Design Activity*, New York, NY: Wiley & Sons, pp. 1-16.

Diaper, D. (1986). Identifying the knowledge requirements of an expert system's natural language processing interface, in Harrison, M., Monk A. (Eds.), *People and computers: designing for usability*, Cambridge University Press, pp. 263-280.

Diaper, D. (Ed.). (1989). *Knowledge elicitation: Principles, techniques and applications*, Chicester, England: Ellis Horwood Ltd.

Davis, Randall. (1979). Interactive Transfer of Expertise: Acquisition of New Inference Rules, In Buchanan, B., Wilkins, D. (Ed.), *Readings in Knowledge Acquisition and Learning,* California: Morgan Kauffman, pp. 221-239.

Eshelman, L., Ehret, D., McDermott, J., Tan, M. (1987). MOLE: a tenacious KA tool, *International Journal of Man-Machine Studies,* 26, pp. 41-54.

Ericsson, K.A., Simon, H.A. (1984). *Protocol Analysis: Verbal Reports as Data*, Cambridge, MA: The MIT Press.

Flanagan, D. (1997a). *JavaScript The Definitive Guide*, California: O'Reilly & Associates, pp. 249-264.

Flanagan, D. (1997b). *JAVA Examples in a Nutshell*, California: O'Reilly & Associates, pp. 104-138.

Flanagan, D. (1997c). *JAVA In A Nutshell*, California: O'Reilly & Associates.

Flanagan, J. (1954). The critical incident technique, *Psychological Bulletin,* 51, pp. 327-358.

Fransella, F., Bannister, D. (1977). *A manual for repertory grid technique,* London: Academic Press.

Gane, C., Sarson, T. (1977). Structured Systems Analysis:--Tools and Techniques, Unpublished document, McDonnell Douglas Corporation.

Garcia, A., Howard, H., Stefik. M. (1993). Active Design Documents: A New Approach for Supporting Documentation in Preliminary Routine Design, Tech. Report 82, Stanford Univ. Center for Integrated Facility Engineering, Stanford, CA.

Geiwitz, J., Kornell, J., McCloskey, B. (1990). An Expert System for the Selection of Knowledge Acquisition Techniques, Technical Report 785-2, Contract No. DAAB07-89-C-A044. California: Anacapa Sciences.

Geiwitz, J., Klatzky, R., McCloskey, B. (1988). Knowledge acquisition techniques for expert systems: Conceptual and empirical comparisions. Santa Barbara, CA: Anacapa Sciences, Inc.

Gero, J. (Winter 1990). Design Prototypes: Knowledge Representation Schema for Design, *AI Magazine*, pp. 26 - 36.

Gordon, S. E., Schmierer, K. A., & Gill, R. T. (1993). Conceptual graph analysis: Knowledge acquisition for instructional system design, *Human Factors*, 35, pp. 459-481.

Gowin, R., Novak, J.D. (1984). *Learning how to learn*, New York: Cambridge University Press.

Gruber (1989). *The Acquisition of Strategic Knowledge*, San Diego, CA: Academic Press.

Hart, A. (1986). *Knowledge acquisition for expert systems*, London: Kogan Page.

Hudlicka, E. (1997). Summary of Knowledge Elicitation Techniques for Requirements Analysis, Course Material for "Human Computer Interaction", Worcester Polytechnic Institute.

Hura, G. S. (1987). Petri net applications, *IEEE Potentials*, October, pp. 25-28.

Johnson, L, Johnson, N. (1987). Knowledge elicitation involving teachback interviewing, in Kidd, A. (Ed.), *Knowledge acquisition for expert systems: a practical handbook*, London: Pitman Press.

Kagel, A. S. (1986). The unshuffle algorithm, *Computer Language*, 1(11), pp. 61-66.

Kelly, G. (1955). *The Psychology of Personal Constructs*, New York: Norton.

Klein, G. A., Calderwood, R., Clinton-Cirocco, A. (1986). Rapid decision making on the fireground, *Proceedings of the 30th Annual Human Factors Society*, 1, Dayton, OH: Human Factors Society, pp. 576-580.

Kingston, J. (1994). Linking Knowledge Acquisition with CommonKADS Knowledge Representation, Presented at *BCS SGES Expert Systems 1004 Conference*, St. John's College, Cambridge, Dec. 1994.

Kingston, J., Aitken, S. (1997). Eliciting Process Models of Intelligence Analysis, *Report of the HPKB Knowledge Acquisition Sessions of 30/31 October, 1997*.

Leddo, J., Cohen, M. (1988). *Cognitive structure analysis: A method of eliciting expert knowledge.* Lexandria, VA: Army Research Institute.

Leddo, J., Mullin, T., Cohen, M., Bresnick, T., Marvin, F., O'Connor, M. (1988). *Knowledge elicitation: Phase I final report*, Vol. 1, Technical Report 87-15, Alexandria, VA: Army Research Institute.

Liu J., Brown D. (1994). Generating Design Decomposition Knowledge for Parametric Design Problems, *Proceedings of AID-94*, Kluwer Academic Publishers, pp. 661-678.

Maher, M. (Winter 1990). Process Models for Design Synthesis, *AI Magazine*, pp. 49 - 58.

Marcus, S., McDermott, J. (1989). SALT: A knowledge acquisition language for propose-and-revise systems, *Artificial Intelligence*, 39, pp. 1-37.

McCloskey, B., Geiwitz, J., Kornell, J. (1991). Empirical Comparisions of Knowledge Acquisition Techniques, *Proceedings of the Human Factors Society 35th Annual Meeting*, Santa Monica, CA: Human Factors Society, pp. 268 - 272.

McGraw K, Harbison-Briggs, K. (1989). *Knowledge Acquisition Principles and Guidelines*, New Jersey: Prentice Hall.

McNeese, M. D., Zaff, B. S. (1991). Knowledge as design: A methodology for overcoming knowledge acquisition bottlenecks in intelligent interface design, *Proceedings of the Human Factors Society 35th Annual Meeting*, Santa Monica, CA: Human Factors Society, pp. 1181-1185.

Miller, P. (1984). *A critiquing approach to expert computer advice: ATTENDING*, Research Notes in Artificial Intelligence 1, London: Pitman Advanced Publishing Program.

Munsen, M. (1998) Protégé, http://smi-web.stanford.edu/projects/protege/, Knowledge Modeling Group, Stanford University School of Medicine.

Nisbett, R., DeCamp Wilson, T. (1977). Telling More Than We Can Know: Verbal Reports on Mental Processes, *Psychological Review*, Vol., 84, No. 3, pp. 231-259.

Olson, J., Biolsi, K. (1990), Techniques for Representing Expert Knowledge, In Ericsson, A., Smith, J. (Eds.) *Toward a General Theory of Expertise*, Cambridge University Press.

OTT (1998). http://www.ott.navy.mil/2_2/2_2_6/, Task Analysis, Chief of Naval

Operations' Office of Training Technology.

Price, S., Kingston, J. (1993). The KADESS Knowledge-Based System: Employing the KADS methodology in an engineering application, Presented at *6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Edinburg, 1-4 June 1993.

Riekert, W. (1991). Knowledge acquisition as an object-oriented modeling process, In M. J. Tauber and D. Ackermann (Eds.) *Mental models and human computer interactions*, Amsterdam: Elsevier Sciences Publishers B. V., pp. 373-381.

Shakeri, C. (1998). Discovery of Design Methodologies for the Integration of Multi-disciplinary Design Problems, Ph.D. Dissertation, ME Department, Worcester Polytechnic Institute.

Shaw, L., Gaines, B. (1995). Comparing Constructions through the Web, *Proceedings of Computer Supported Cooperative Learning*. Bloomington, October, 1995.

Shaw, L., Gaines, B. (1998). WebGrid II: Developing Hierarchical Knowledge Structures from Flat Grids, *Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling, and Management*, Banff, Alberta, Canada, April, 1998.

Shute, V. J. (1998). DNA: Towards an Automated Knowledge Elicitation and Organization Tool, submitted for *Cognitive Tools-II.*

Smith, R., Tjandra, P. (1998). Experimental Observation of Iteration in Engineering Design, *Research in Engineering Design*, Vol. 10, No. 2.

Smithers, T. (1998). Towards a Knowledge Level Theory of Design Process, *Proceedings of AID-98*, Kluwer Academic Publishers.

Spradley, J. (1980). *Participant observation*, London: Holt, Rienhart, & Winston.

Spradley, J. (1979). *The ethnographic interview,* London: Holt, Rienhart, & Winston.

Swaffield, G., Knight, B. (1990). Applying system analysis techniques to knowledge engineering, *Expert Systems*, 1, pp. 82-93.

Thordsen, M. (1991). A Comparison of Two Tools for Cognitive Task Analysis: Concept Mapping and the Critical Decision Method, *Proceedings of the Human Factors Society 35th Annual Meeting*.

Van de Velde, W. (1994). An Overview of CommonKADS. In J. Brueker, W. Van de Velde (Ed.) *CommonKADS Library for Expertise Modeling*, Amsterdam: IOS Press, pp. 9-30.

Weingaertner, S. T., Lewis, A. H. (1988). Evaluation of decision aiding in submarine emergency decision making, In J. Ranta (Ed.) *Analysis, Design, and Evaluation of Man-Machine Systems: Selected Papers from the 3rd IFAC/IEA/IFORS Conference*, 1, Oxford, UK: Pergamon, pp. 95-201.

Whaley, C. P. (1979). Collecting paired-comparison data with a sorting algorithm, *Behavior Research Methods and Instrumentation*, 11, pp. 147-150.

Witt, G. (1998). A Comparison of Knowledge Elicitation Techniques for Describing Conceptual Knowledge in Declarative and Procedural Domains, Ph.D. Dissertation, Psychology Department, George Mason University

Woods, D. D., Hollnagel, E. (1987). Mapping cognitive demands in complex problem-solving worlds, *International Journal of Man-Machine Studies*, 26, pp. 257-275.

Appendix A Task Stimuli

This appendix contains the data file that specifies where the task stimuli is located and the task stimuli themselves.

A.1 Task Stimuli Data File

Design Problem Description
http://cs.wpi.edu/~jburge/DOES/design_problem.html
Design Task Description
http://cs.wpi.edu/~jburge/DOES/design_task.html
Design Step Examples
http://cs.wpi.edu/~jburge/DOES/design_step.html

A.2 Design Problem Description

Airport Database Design

The overall problem is to design an airport database system.

From [Ramakrishnan, "Database Management Systems", 1997]:

Airport Database Requirements:

- Every airplane has a registration number, and each airplane is of a specified model.
- The airport accommodates a number of airplane models, and each model is identified by a model number (e.g., DC-10) and has a capacity and a weight.
- A number of technicians work at the airport. You need to store the name, SSN, address, phone number, and salary of each technician.
- Each technician is an expert on one or more plane model(s), and his or her experience may overlap with that of other technicians. This information about technicians must also be recorded.
- Traffic controllers must have an annual medical examination. For each traffic controller, you must store the date of the most recent exam.
- All airport employees (including technicians) belong to a union. You must store the union membership number of each employee. You can assume that each employee is uniquely identified by the social security number.

- The airport has a number of tests that are used periodically to ensure that airplanes are still airworthy. Each test has a Federal Aviation Authority (FAA) test number, a name, and a maximal possible score.
- The FAA requires the airport to keep track of each time that a given airplane is tested by a given technician using a given test. For each testing event, the information needed is the date, the number of hours the technician spent doing the test, and the score that the airplane received on the test.

A.3 Design Task Description

Design Task Instructions

Please describe how you would design an ER diagram for the airport database. Be sure to describe the various attributes of each entity and relationship set; also describe the key and participation constraints for each relationship set. Describe any necessary overlap and covering constraints as well.

A.4 Design Step Example

Design Step Example

Design steps should be the "elementary" tasks and decisions made during the design process.

Examples of "elementary" tasks for ER diagram design would be:

For entity creation:

Create an [entity name] entity with attributes [attribute list].

For relationship creation:

Create a one-to-one relationship, [relationship name] between [entity 1] and [entity 2] where [entity 1] [relationship name] one-and-only-one [entity 2].

The steps listed should be specific enough that an entity relationship diagram could be drawn from them without looking at the original
problem description.

Appendix B Sample Output from DOES

This appendix contains examples of DOES output files. All names have been removed to maintain the anonymity of the subjects.

B.1 Usability Survey Results

```
Background Information

Subject Name:
  --name--

Email address:
  --email--

Educational Background:
  MS

Relevant Experience:
  1-5

Relevant Training:
  -one year employment at a research company in Germany (systems design)

Relevant Projects:
  -software purchase database system for a university CS dept.
-all-purpose information management system for engineering companies
-workflow system for engineering companies
```

B.2 Knowledge Elicitation Experiment Results

```
Design Description:
  1) Identify the relations (entities)
2) Identify the attributes for each relation
3) Identify the primary key for each relation
4) Identify the relationships between the tables
```

5) Noramlize - reduce redundancy
6) Identify the foreign keys

Initial Steps:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
Create an entity Model with attributes (ModelNumber, capacity, weight)
Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)
Create an entity TechExpert with attributes (SSN, ModelNumber)
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Employees with attributes (SSN, unionMembershipID)
create an entity Test with attributes (FAANumber, Name, MaxScore)
create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
create a m-to-n relationship TechExpert bewteen Technician and Model
create a m-to-m relationship TestingEvent between Test and Airplane
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
createcreate primary key Airplane.RegistrationNumber
create primary key Model.ModelNumber
create primary key Technician.SSN
create primary key Test.FAANumber

Sorted Steps:

Create an entity Model with attributes (ModelNumber, capacity, weight)
create primary key Model.ModelNumber
create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
createcreate primary key Airplane.RegistrationNumber

create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)
create primary key Technician.SSN
create a m-to-n relationship TechExpert bewteen Technician and Model
Create an entity TechExpert with attributes (SSN, ModelNumber)
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Employees with attributes (SSN, unionMembershipID)
create an entity Test with attributes (FAANumber, Name, MaxScore)
create primary key Test.FAANumber
create a m-to-m relationship TestingEvent between Test and Airplane
create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber

Current Step:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)

Prior:

Others:

Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)
Create an entity TechExpert with attributes (SSN, ModelNumber)
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Employees with attributes (SSN, unionMembershipID)
create an entity Test with attributes (FAANumber, Name, MaxScore)
create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)

create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
create a m-to-n relationship TechExpert bewteen Technician and Model
create a m-to-m relationship TestingEvent between Test and Airplane
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
createcreate primary key Airplane.RegistrationNumber
create primary key Model.ModelNumber
create primary key Technician.SSN
create primary key Test.FAANumber
Create an entity Model with attributes (ModelNumber, capacity, weight)

Current Step:

Create an entity Model with attributes (ModelNumber, capacity, weight)

Prior:

Others:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)
Create an entity TechExpert with attributes (SSN, ModelNumber)
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Employees with attributes (SSN, unionMembershipID)
create an entity Test with attributes (FAANumber, Name, MaxScore)
create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
create a m-to-n relationship TechExpert bewteen Technician and Model

create a m-to-m relationship TestingEvent between Test and Airplane
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
createcreate primary key Airplane.RegistrationNumber
create primary key Model.ModelNumber
create primary key Technician.SSN
create primary key Test.FAANumber

Current Step:

Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)

Prior:

Others:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
Create an entity Model with attributes (ModelNumber, capacity, weight)
Create an entity TechExpert with attributes (SSN, ModelNumber)
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Employees with attributes (SSN, unionMembershipID)
create an entity Test with attributes (FAANumber, Name, MaxScore)
create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
create a m-to-n relationship TechExpert bewteen Technician and Model
create a m-to-m relationship TestingEvent between Test and Airplane
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN

cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
createcreate primary key Airplane.RegistrationNumber
create primary key Model.ModelNumber
create primary key Technician.SSN
create primary key Test.FAANumber

Current Step:

Create an entity TechExpert with attributes (SSN, ModelNumber)

Prior:

Create an entity Model with attributes (ModelNumber, capacity, weight)
Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)

Others:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Employees with attributes (SSN, unionMembershipID)
create an entity Test with attributes (FAANumber, Name, MaxScore)
create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
create a m-to-n relationship TechExpert bewteen Technician and Model
create a m-to-m relationship TestingEvent between Test and Airplane
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber

create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
createcreate primary key Airplane.RegistrationNumber
create primary key Model.ModelNumber
create primary key Technician.SSN
create primary key Test.FAANumber

Current Step:

Create an entity TrafficController with attributes (SSN, DateOfExam)

Prior:

Others:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
Create an entity Model with attributes (ModelNumber, capacity, weight)
Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)
Create an entity TechExpert with attributes (SSN, ModelNumber)
create an entity Employees with attributes (SSN, unionMembershipID)
create an entity Test with attributes (FAANumber, Name, MaxScore)
create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
create a m-to-n relationship TechExpert bewteen Technician and Model
create a m-to-m relationship TestingEvent between Test and Airplane
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber

createcreate primary key Airplane.RegistrationNumber
create primary key Model.ModelNumber
create primary key Technician.SSN
create primary key Test.FAANumber

Current Step:

create an entity Employees with attributes (SSN, unionMembershipID)

Prior:

Others:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
Create an entity Model with attributes (ModelNumber, capacity, weight)
Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)
Create an entity TechExpert with attributes (SSN, ModelNumber)
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Test with attributes (FAANumber, Name, MaxScore)
create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
create a m-to-n relationship TechExpert bewteen Technician and Model
create a m-to-m relationship TestingEvent between Test and Airplane
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
createcreate primary key Airplane.RegistrationNumber
create primary key Model.ModelNumber
create primary key Technician.SSN
create primary key Test.FAANumber

Current Step:

create an entity Test with attributes (FAANumber, Name, MaxScore)

Prior:

Others:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
Create an entity Model with attributes (ModelNumber, capacity, weight)
Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)
Create an entity TechExpert with attributes (SSN, ModelNumber)
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Employees with attributes (SSN, unionMembershipID)
create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
create a m-to-n relationship TechExpert bewteen Technician and Model
create a m-to-m relationship TestingEvent between Test and Airplane
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
createcreate primary key Airplane.RegistrationNumber
create primary key Model.ModelNumber
create primary key Technician.SSN
create primary key Test.FAANumber

Current Step:

create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)

Prior:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
create an entity Test with attributes (FAANumber, Name, MaxScore)
Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)

Others:

Create an entity Model with attributes (ModelNumber, capacity, weight)
Create an entity TechExpert with attributes (SSN, ModelNumber)
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Employees with attributes (SSN, unionMembershipID)
create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
create a m-to-n relationship TechExpert bewteen Technician and Model
create a m-to-m relationship TestingEvent between Test and Airplane
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
createcreate primary key Airplane.RegistrationNumber
create primary key Model.ModelNumber
create primary key Technician.SSN
create primary key Test.FAANumber

Current Step:

create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber

Prior:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
Create an entity Model with attributes (ModelNumber, capacity, weight)

Others:

Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)
Create an entity TechExpert with attributes (SSN, ModelNumber)
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Employees with attributes (SSN, unionMembershipID)
create an entity Test with attributes (FAANumber, Name, MaxScore)
create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
create a m-to-n relationship TechExpert bewteen Technician and Model
create a m-to-m relationship TestingEvent between Test and Airplane
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
createcreate primary key Airplane.RegistrationNumber
create primary key Model.ModelNumber
create primary key Technician.SSN
create primary key Test.FAANumber

Current Step:

create a m-to-n relationship TechExpert bewteen Technician and Model

Prior:

Create an entity Model with attributes (ModelNumber, capacity, weight)
Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)

Others:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
Create an entity TechExpert with attributes (SSN, ModelNumber)
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Employees with attributes (SSN, unionMembershipID)
create an entity Test with attributes (FAANumber, Name, MaxScore)
create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
create a m-to-m relationship TestingEvent between Test and Airplane
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
createcreate primary key Airplane.RegistrationNumber
create primary key Model.ModelNumber
create primary key Technician.SSN
create primary key Test.FAANumber

Current Step:

create a m-to-m relationship TestingEvent between Test and Airplane

Prior:

Create an entity Model with attributes (ModelNumber, capacity, weight)
create an entity Test with attributes (FAANumber, Name, MaxScore)

Others:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)

Create an entity TechExpert with attributes (SSN, ModelNumber)
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Employees with attributes (SSN, unionMembershipID)
create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
create a m-to-n relationship TechExpert bewteen Technician and Model
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
createcreate primary key Airplane.RegistrationNumber
create primary key Model.ModelNumber
create primary key Technician.SSN
create primary key Test.FAANumber

Current Step:

create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN

Prior:

create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)

Others:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
Create an entity Model with attributes (ModelNumber, capacity, weight)
Create an entity TechExpert with attributes (SSN, ModelNumber)
Create an entity TrafficController with attributes (SSN, DateOfExam)

create an entity Employees with attributes (SSN, unionMembershipID)
create an entity Test with attributes (FAANumber, Name, MaxScore)
create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
create a m-to-n relationship TechExpert bewteen Technician and Model
create a m-to-m relationship TestingEvent between Test and Airplane
cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
createcreate primary key Airplane.RegistrationNumber
create primary key Model.ModelNumber
create primary key Technician.SSN
create primary key Test.FAANumber

Current Step:

cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber

Prior:

create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
create an Airplane entity with attributes (RegistrationNumber, ModelNumber)

Others:

Create an entity Model with attributes (ModelNumber, capacity, weight)
Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)
Create an entity TechExpert with attributes (SSN, ModelNumber)
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Employees with attributes (SSN, unionMembershipID)
create an entity Test with attributes (FAANumber, Name, MaxScore)
create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber

create a m-to-n relationship TechExpert bewteen Technician and Model
create a m-to-m relationship TestingEvent between Test and Airplane
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
createcreate primary key Airplane.RegistrationNumber
create primary key Model.ModelNumber
create primary key Technician.SSN
create primary key Test.FAANumber

Current Step:

create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber

Prior:

create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
create an entity Test with attributes (FAANumber, Name, MaxScore)

Others:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
Create an entity Model with attributes (ModelNumber, capacity, weight)
Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)
Create an entity TechExpert with attributes (SSN, ModelNumber)
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Employees with attributes (SSN, unionMembershipID)
create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
create a m-to-n relationship TechExpert bewteen Technician and Model
create a m-to-m relationship TestingEvent between Test and Airplane
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN

cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
createcreate primary key Airplane.RegistrationNumber
create primary key Model.ModelNumber
create primary key Technician.SSN
create primary key Test.FAANumber

Current Step:

create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN

Prior:

Create an entity TechExpert with attributes (SSN, ModelNumber)
Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)

Others:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
Create an entity Model with attributes (ModelNumber, capacity, weight)
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Employees with attributes (SSN, unionMembershipID)
create an entity Test with attributes (FAANumber, Name, MaxScore)
create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
create a m-to-n relationship TechExpert bewteen Technician and Model
create a m-to-m relationship TestingEvent between Test and Airplane
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber

create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
createcreate primary key Airplane.RegistrationNumber
create primary key Model.ModelNumber
create primary key Technician.SSN
create primary key Test.FAANumber

Current Step:

create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber

Prior:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
Create an entity TechExpert with attributes (SSN, ModelNumber)

Others:

Create an entity Model with attributes (ModelNumber, capacity, weight)
Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Employees with attributes (SSN, unionMembershipID)
create an entity Test with attributes (FAANumber, Name, MaxScore)
create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
create a m-to-n relationship TechExpert bewteen Technician and Model
create a m-to-m relationship TestingEvent between Test and Airplane
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
createcreate primary key Airplane.RegistrationNumber
create primary key Model.ModelNumber

create primary key Technician.SSN
create primary key Test.FAANumber

Current Step:

createcreate primary key Airplane.RegistrationNumber

Prior:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)

Others:

Create an entity Model with attributes (ModelNumber, capacity, weight)
Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)
Create an entity TechExpert with attributes (SSN, ModelNumber)
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Employees with attributes (SSN, unionMembershipID)
create an entity Test with attributes (FAANumber, Name, MaxScore)
create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
create a m-to-n relationship TechExpert bewteen Technician and Model
create a m-to-m relationship TestingEvent between Test and Airplane
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
create primary key Model.ModelNumber
create primary key Technician.SSN
create primary key Test.FAANumber

Current Step:

create primary key Model.ModelNumber

Prior:

Create an entity Model with attributes (ModelNumber, capacity, weight)

Others:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)
Create an entity TechExpert with attributes (SSN, ModelNumber)
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Employees with attributes (SSN, unionMembershipID)
create an entity Test with attributes (FAANumber, Name, MaxScore)
create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
create a m-to-n relationship TechExpert bewteen Technician and Model
create a m-to-m relationship TestingEvent between Test and Airplane
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
createcreate primary key Airplane.RegistrationNumber
create primary key Technician.SSN
create primary key Test.FAANumber

Current Step:

create primary key Technician.SSN

Prior:

Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)

Others:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
Create an entity Model with attributes (ModelNumber, capacity, weight)
Create an entity TechExpert with attributes (SSN, ModelNumber)
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Employees with attributes (SSN, unionMembershipID)
create an entity Test with attributes (FAANumber, Name, MaxScore)
create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
create a m-to-n relationship TechExpert bewteen Technician and Model
create a m-to-m relationship TestingEvent between Test and Airplane
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
createcreate primary key Airplane.RegistrationNumber
create primary key Model.ModelNumber
create primary key Test.FAANumber

Current Step:

create primary key Test.FAANumber

Prior:

create an entity Test with attributes (FAANumber, Name, MaxScore)

Others:

create an Airplane entity with attributes (RegistrationNumber, ModelNumber)
Create an entity Model with attributes (ModelNumber, capacity, weight)
Create an entity Technician with attributes (Name, SSN, address, phoneNumber, salary)
Create an entity TechExpert with attributes (SSN, ModelNumber)
Create an entity TrafficController with attributes (SSN, DateOfExam)
create an entity Employees with attributes (SSN, unionMembershipID)
create an entity TestingEvent with attributes (TechID, AirplaneRegistrationNumber, date, NumberOfhours, score)
create a m-to-one relationship, airPlaneModel, between Airplane and Model where Airplane.ModelNumber = Model.ModelNumber
create a m-to-n relationship TechExpert bewteen Technician and Model
create a m-to-m relationship TestingEvent between Test and Airplane
create a n-to-1 relationship tech between TestingEvent and Technician where TestingEvent.TechnicianID = Technician.SSN
cretae a n-to-1 relationship model between TestingEvent and Airplane where TestingEvent.RegistrationNumber = Airplane.RegistrationNumber
create a n-to-1 relationship test between Test and TestingEvent where TestingEvent.FAANumber = Test.FAANumber
create a n-to-1 relationship techExpert between TechExpert and Technician where TechExpert.SSN = Technician.SSN
create a n-to-1 relationship techModel between TechExpert and Airplane where TechExpert.ModelNumber = Airplane.ModelNumber
createcreate primary key Airplane.RegistrationNumber
create primary key Model.ModelNumber
create primary key Technician.SSN

B.3 Usability Survey Results

Usability Survey Results:

Usability:
easy

Data sufficient:
ok

Time it took was:
31-60 minutes

Made me think:
yes

Better explanation of design:
yes

Comments:
Design Process: I found myself trying to order steps and modify/merge/decompose entities/relations at the onset (question 1). Consequently, I'm not
sure if the information I provided is what you were really looking for - unless, of course, this was part of the experiment. Not much moved around in
terms of ordering steps.

UI: Having wider text entry controls would make it easier to see and select a desired entity/relationship.