Pergamon

Information Processing & Management, Vol. 31, No. 1, pp. 47-58, 1995

0306-4573/95 $9.50 + .00

0306-4573(94)00033-6

INFORMATION/KNOWLEDGE ACQUISITION METHODS FOR DECISION SUPPORT SYSTEMS AND EXPERT SYSTEMS

HENG-LI YANG
Department of Management Information Systems, College of Business Administration, National Cheng-Chi University, 64, Sec.2, Chihnan Road, Taipei, Taiwan
E-mail: yanh@mis.nccu.edu.tw

(*Received* 18 *November* 1993; *accepted in final form* 16 *May* 1994)

Abstract—The construction of a decision support system must define its information requirements. Building an expert system needs knowledge acquisition—the accumulation, transfer, and transformation of problem-solving expertise from some knowledge source. There are a few methods associated with the information requirement elicitation (IRE) and knowledge acquisition (KA). This paper compares representative IRE and KA methods. It concludes that though most methods can be applied to both types of systems, there are still few methods that are unique to one type.

Keywords: Decision support systems, Expert systems, Information requirement elicitation, Knowledge acquisition.

1. INTRODUCTION

The effectiveness of an information system requires the thorough elicitation of user information requirements. Similarly, a key aspect of expert systems development is knowledge acquisition (KA). Knowledge acquisition is the general name given to the process of eliciting, acquiring, and representing knowledge consisting of descriptions, relationships, and procedures in a specialized domain of interest (Dhaliwal & Benbasat, 1989). Its major functions are to extract knowledge from expert(s), and analyze and formalize the knowledge into some computer understandable forms (Shaw & Woodward, 1989). It should appropriately represent the knowledge structure to conceptualize the mental model(s) or expert(s) and facilitate inference and explanation capabilities.

Byrd *et al.* (1992) compare some information requirement-elicitation (IRE) and knowledge-acquisition methods. They demonstrate that these two research streams have many things in common, and techniques from IRE and KA address very similar issues. However, it is doubtful whether all methods can be applied to any information system development.

The focus of this paper is on two special types of information systems—decision support systems (DSS) and expert systems (ES). In fact, DSS and ES are as ends of a continuum. There are some efforts to integrate these two systems into one, and the distinction between these two systems has become fuzzy. However, this paper adopts the traditional views of these two systems for comparison. IRE methods for DSS development are compared with KA methods for ES development. The purpose is neither to survey all available methods nor to provide the details of these methods. Rather, the applicability of the different methods is discussed from the point of view of the *different characteristics* underlying these two types of systems.

The major intention of this paper is to test two extreme hypotheses: (1) all IRE methods and KA methods are unique to DSS and ES development, respectively; that is, there is no common method, and (2) all methods are common. Hypothesis I may be trivially rejected by the conclusion of the previous paper (Byrd *et al.* 1992). The testing of Hypothesis II is more interesting.

If some methods can be found to reject Hypothesis II, it is worth discussing why—

does the difference of the underlying system characteristics cause DSS and ES to need different development methods?

In the following, the definition and architectures of ES and DSS are first reviewed and compared in Section 2. Section 3 discusses the development cycles and IRE/KA methods for two types of systems. Section 4 discusses how and why the methods used in ES development differ from those used in DSS development, and the applicability of methods to both systems. Section 5 concludes this paper.

2. OVERVIEW OF DSS AND ES

2.1 *Definitions*

2.1.1 *DSS.* It is somewhat difficult to articulate the definition of DSS because there is no consensus on what a DSS is. One definition, consistent with those of Turban (1993) and Sprague & Carlson (1982), is: A DSS is an interactive, computer-based information system that utilizes decision rules and models, coupled with a comprehensive database to support all phases of the decision-making process mainly in semistructured (or unstructured) decisions under the full control of the decision makers. Here the term "semistructured decision" is used for situations in which information needs cannot be described in detail before making a decision, and the procedures for obtaining the best (or a good enough) solution are unknown. In addition, the following characteristics and capabilities are helpful to check whether a system is a DSS or not.

- DSSs support all phases of the decision-making process: intelligence, design, choice, and implementation (Turban, 1993).
- Support is provided for various managerial levels, ranging from top executives to line managers, and to individuals as well as to groups.
- DSSs support several interdependent and/or sequential decisions and a variety of decision-making processes and styles.
- DSSs must be easy to use and adaptive over time. DSSs are so flexible that the decision maker is able to confront changing conditions quickly and adapt the DSSs to meet these changes.
- DSSs attempt to improve the effectiveness of decision making, rather than its efficiency.

2.1.2 *ES.* According to Turban (1993), an expert system is a computerized advisory program that attempts to imitate or substitute the reasoning processes and knowledge of experts in solving specific type of problems. More precisely, it solves real-world problems requiring an expert's interpretation, employs heuristic knowledge and/or qualitative models of the problem domain, reaches the same conclusions that a human expert would have if faced with a similar situation, and employs a programming methodology based on a separation between knowledge and its application (Pfeifer & Lüthi, 1987).

Human experts are scarce and expensive. The objective of an expert system is the dissemination of expertise: transferring expertise from some expert(s) to a computer and then on to other human nonexperts. The problem areas addressed by expert systems include interpretation, predication, diagnosis, design, planning, monitoring, debugging, repair, instruction, and control (Turban, 1993).

2.1.3 *General architectures of DSS and ES.* Three major components of a DSS are:

- *data management component* for relevant data,
- *model management component* for analytical capabilities, and
- *communication (dialogue) subsystem* through which the user can communicate with the DSS.

An expert system is composed of:

- *knowledge base* for a particular problem domain,
- *inference engine* to propagate inferences over the knowledge,
- *explanation subsystem* to explain what/why/how about the reasoning process,
- *user interface* for problem-oriented communications between the user and computer, and
- *workplace* (i.e., an area of working memory) for the description of a current problem.

2.1.4 *Comparison of DSS and ES. Similarity:* ES and DSS share some characteristics. As Turban (1993) pointed out, one major similarity is to give assistance in solving unstructured problems. According to Luconi *et al.* (1986), no straightforward solution techniques (e.g., standard procedures) for such problems are known, and so some flexible problem-solving strategies are used to decide which standard procedures (i.e., algorithms or formulas) to apply to solve problems.

Difference: If we adopt the traditional views of these two types of systems, there are some fundamental differences between them. Here, the differences are examined in terms of the following dimensions.

1. *Origination of system.* Historically, DSSs are mainly from the discipline of management science and decision science to improve (management) decision making, whereas ESs are from the discipline of artificial intelligence for problem solving in a particular domain.

2. *Objective of system.* As has been pointed out by a number of authors (e.g., Keen & Scott-Morton, 1978; Sprague, 1980) a task (such as management decision making) that is at the same time multi-dimensional, multi-objective, and only partially defined, cannot be automated. One reason is that such tasks typically employ much "common sense." So the objective of a DSS is not to automate management decision making, but rather to *support the intuition of the decision maker.* On the other hand, the objective of an ES is to *replicate a human advisor and to replace him or her.*

3. *Characteristics of problem area.* The problems most DSSs deal with are broad and ill specified (i.e., the goals are typically formulated in very general and vague ways; only partial specifications are available)—for example, the new product or service planning. The applicability of ES is restricted to narrow domain and (more or less) well defined problems—for example, the car repair problem. That is because human experts are good only if they operate in a very narrow domain. However, the fact that a problem is well defined does not mean it is easy to solve, because the solution path may be highly complex (Pfeifer & Lüthi, 1987).

4. *Type of problem treated.* DSSs are more suitable for dealing with *ad hoc* and unique situations, whereas ESs are more suitable for providing advice on repetitive problem areas. The repetitiveness of problems may indicate the high payoff from ESs.

5. *Representation of problem solving process.* A DSS has a sparse representation of the decision-making process. It only contains an implicit (and typically rather vague) model of the decision process involved. On the other hand, an ES has a dense representation of the problem-solving process. It contains an implicit or explicit model of the kind of information needed and how this information is processed. For example, in a car repair ES, though the system might not have a basic physics or engineering model, it contains all the necessary information to repair a car—the mental model of a human repair expert (if the system is appropriately constructed), and "knows" what kinds of input data it needs.

6. *Control and interactivity (possibility for user intervention).* A DSS must allow the user to confront a problem in a flexible and personal way by providing the ability to manipulate the data and models in a variety of ways through the decision-making process. So, *the initiative is always with the user.* However, an ES contains a more or less complete model of the problem-solving process. With minor exceptions, *the initiative is always with the system*—the system asks for data, performs all the reasoning steps, asks for more information, and in the end comes up with a solution.

In some systems with advanced options (e.g., ESs for medical diagnosis), ESs can be

used to conduct sensitivity analysis (i.e., what-if analysis similar to a DSS) to change any of the input data (e.g., likelihood estimates). The system would ask more questions and then display new recommendation(s). However, basically it is the system rather than the use to take most of the initiative.

7. *Users of system.* A DSS may provide support for managers, ranging from top executives to line managers, whereas an ES has several possible users: a nonexpert client seeking direct advice, a pupil or student who wants to learn, an ES builder who wants to improve or increase the knowledge base, and an expert who wants an assistant.

8. *Types of knowledge in database/knowledge base.* The database of a DSS contains facts (declarative knowledge), whereas the knowledge base of an ES contains facts, procedural knowledge, heuristics, and others (e.g., judgments, causal knowledge, meta-knowledge).

9. *Reasoning capability.* An ES, by definition, exhibits some reasoning capability. A DSS does not possess such a capability at all (though an advanced DSS, called intelligent DSS, may include knowledge subsystem and incorporate reasoning capabilities).

10. *Explanation capability.* A DSS user can normally trust the quality of information produced by the individual modules of the DSS. The whole process is under his or her control and he or she can use the modules or not. However, an ES user plays a passive role in the problem-solving process and gets a solution. Is the solution given by the system to be trusted without question? This necessitates an explanatory capability of the system to allow users to decide whether to trust the solution or not.

11. *Source of knowledge.* DSSs mainly use knowledge from the *reconstructed* sources (e.g., statistical, management science, or quantitative models) to provide the system's analytical capabilities, whereas ESs can use knowledge from *authentic* sources (the chosen human experts) as well as the *reconstructed* sources.

12. *Nature of support.* DSSs can provide *individual, group, and organizational* supports. Group support is to a group of people, each engaged in separate but highly interrelated tasks. Organizational support is to organizational tasks or activities involving a sequence of operations, different functional areas, and required resources. ESs mainly give advice and explanations to *an individual (or a group)*.

In summary, because these two types of systems have different objectives, their representation of the problem-solving process, initiative, incorporated knowledge, etc., are different. The different representation of the problem-solving process would have great impact on their development methods, as shown in later sections.

3. INFORMATION/KNOWLEDGE ACQUISITION METHODS IN DSS/ES DEVELOPMENT

3.1 *System development cycles*

Turban (1993) summarized a DSS development life cycle as: predesign, design, construction, implementation, maintenance, and adaptation. Predesign stage is just like information requirements and analysis phase in a traditional system development life cycle (SDLC). Similarly, there is also a SDLC in an ES. Benbasat & Dhaliwal (1989) have proposed an ES development life cycle as: modelling, knowledge elicitation, system construction, experiment, and use.

Comparing the development cycles of DSS and ES, we observe that the knowledge acquisition in ES development is a phase similar to information requirements and analysis stage in DSS development. Each of their functions is to determine what knowledge should be put into the systems, though there are some substantial differences in the knowledge contents, knowledge source, and acquisition processes. In a DSS, information requirements are from users to system builders. Then the system builders design a system to meet the requirements. In an ES, the system builder must extract the knowledge from the expert. The user requirements (e.g., graphic interface and explanation facilities) are generally assumed clear or are simply ignored, though some researchers emphasize their importance in developing ESs (Breuker & Wielinga, 1987; Greenwell, 1988).

3.2 *Information/knowledge acquisition methods in DSS/ES*

This section surveys some methods often mentioned in IRE or KA research area. The description for each method is brief. As mentioned in Section 1, the purpose is not to survey all available methods. Rather, the intention is to have some sample methods to test the following extreme hypotheses:

HYPOTHESIS I

There is no common method to DSS and ES development. That is, each IRE or KA method is unique to DSS or ES, respectively.

HYPOTHESIS II

All development methods are common to DSS and ES. That is, any IRE or KA method can be applied to either system development.

3.2.1 *DSS information requirement elicitation. ROMC:* ROMC is a process-independent method developed by Sprague & Carlson (1982). It assumes that a DSS should support the commonality of decision making and leave the users themselves to develop their own styles, knowledge, and skills. This method is intended to identify information requirements in each of three capability areas (intelligence, design, and choice) of DSS. One set of identified capabilities can support a variety of decision-making processes. Four components constitute this method.

- *Representations:* provide representations (e.g., pictures, chart, number, equation, table, report, etc.) to help the decision maker conceptualize and communicate the problem, since decision makers rely on these kinds of conceptualizations in making or explaining a decision.
- *Operations:* provide operations to analyze and manipulate those representations.
- *Memory aids:* provide a set of memory aid such as databases, libraries, etc., to assist the user in linking representations and operations.
- *Control Mechanisms:* facilitate users control and operation of the entire system and integrate the representations, operations, and memory aids into useful decision-making systems that would fit the users knowledge, skills, and style.

One should note that the representations are not as detailed as the decision makers' conceptualizations, and the operations might support only common decision-making activities (Sprague & Carlson, 1982). It is a *process-independent* approach, and encourages decision makers to exercise direct, personal control over the support (e.g., let the decision maker select the sequencing of operations).

Asking: Using this method, the system analyst directly asks users the information requirements either by interviews or questionnaires with closed/open questions, or through brainstorming, guided brainstorming, or group consensus. These methods can be used in determining both organizational and specific application information requirements (Davis, 1982).

Deriving from existing information systems: The existing systems that have been implemented can be used to derive requirements for a proposed information system for the same type of organizations or for the same type of applications. Textbooks, handbook, industry manuals, and packages can also be used to determine the requirements. This method is suitable for information requirements determination in a stable environment (Davis, 1982).

Synthesis from characteristics of utilizing systems: Information requirements are originated from the activities of the object systems (the real world, i.e., the organization). Analysis of the characteristics of the utilizing organization is a logical way to get the information requirements for the new system. A lot of methods fall into this category.

1. *Event analysis* (Bahl & Hunt, 1984): Events are identifiable activities in decision making. Decisions can be described by analysis of the events. Endogenous events may consist of activities carried out by actors both inside and outside an organization. Exogenous

events involve activities that are not part of the decision-making process itself, but are considered a relevant part of the environment.

2. *Decision participants analysis* (Bahl & Hunt, 1984): The user positions in the organization and roles in the decision making are related to the decision information requirements. It begins with exhaustively listing all actors who participate or are expected to participate in a decision. By explaining their roles and relationships in the decision process, the information requirements are clearer.

3. *Decision process and contents analysis* (Davis, 1982; Bahl & Hunt, 1984): The whole process can be divided into three periods: pre-decision, decision, and post-decision. It begins with identifying a decision, defines its decision process by means of some tools (e.g., decision tables), and then finds out information needed for the decision process.

4. *Critical success factor (CSF) analysis* (Davis, 1982): CSF analysis is a method of eliciting requirements by asking users to define the factors that are critical to success in performing their functions or making decisions. CSFs are the few key areas that must go right if an organization is to be successful. A series of interviews and meetings can transform individual CSFs into departmental and overall organizational CSFs.

5. *Strategy set transformations* (Davis, 1982; Davis & Olson, 1982; Meador & Rosenfeld, 1986): The organization and decision area's objectives may result in different information requirements. By identifying, validating, and then transforming the organizational goals and strategies, the organizational information requirements can be determined.

6. Business systems planning (BSP) (David, 1982): BSP is a comprehensive IBM methodology. Information requirements are derived from the organization in a top-down fashion by starting with business objectives and then defining business processes. Business processes are used to identify the business entities, data classes, and their relationships. An information architecture is proposed by relating processes to data classes, and the priorities for the development of the applications are then set.

Among these methods, CSF, strategy set transformation, and BSP are more suitable for organizational information requirements determination.

Prototyping: Prototyping as a DSS development approach was introduced mainly because the user requirements are difficult to be predefined or very changeable. It is also very effective for user requirement determination. Users interact with their system in its environment to specify their requirements more completely and correctly (Naumann & Jenkins, 1982; Davis, 1982).

Process tracing methods: In recent literature, it is pointed out that to extend the support areas, DSS research should emphasize decision makers rather than just the decision itself (Bahl & Hunt, 1985). Process tracing methods (Todd & Benbasat, 1987) are different from the above discussed techniques because they focus on the decision-maker behaviors in the decision-making process and try to open the "black box." This set of methods includes information display board, eye movement tracing, computer logs, written protocol, and verbal protocol. It is observed that among them, protocol analysis is the most powerful for discovering the dynamics of problem definitions, hypothesis formation, and information search in a less structured context. This method can be used for identifying biases and inconsistencies of the decision maker in the process, and provide the relevant support to overcome the weaknesses; facilitating designers to be aware of the heuristics, methods, and information sources used to solve a particular problem; and developing an effective user-computer interface.

However, verbal protocol analysis is a controversial technique and has many detractors (e.g., often time-consuming and subject to bias when techniques for eliciting verbalization are not rigorous). A balanced picture appears to show that under specific conditions verbal protocols are valid and do provide data that is not available from other methods (Ericsson & Simon, 1984).

3.2.2 *ES knowledge acquisition methods.* Knowledge acquisition has been the bottleneck in ES development because of human knowledge characteristics, lack of theoretical support, and overall methodological framework for knowledge acquisition, as well as the high demands on knowledge engineer skills (Shaw & Woodward, 1989). Fortunately,

this field has drawn great attention from researchers from different areas, and numerous methods have been developed in recent years.

Direct methods: Direct methods ask the expert to report on knowledge he or she can directly articulate (Olson & Rueter, 1987). These methods include *interviews, questionnaires, observations, protocol analysis, interrupt analysis*, and *inferential flow analysis*. Direct methods rely on the availability of the information to both introspection and articulation. In addition, the knowledge engineer's communication skills also affect the elicitation process.

Indirect methods: It is not uncommon for human experts to perceive complex relationships or come to sound conclusions without knowing exactly how they did it. These indirect methods do not directly ask the experts to express their knowledge. Instead they focus on inferring what the expert "must have known" from his or her responses to the cases. Based on the elicitation, the knowledge engineer infers underlying structures among the objects rated or recalled (Olson & Rueter, 1989; Dhaliwal & Benbasat, 1989). The typical indirect methods are *multidimensional scaling, Johnson hierarchical clustering, general weighted networks, repertory grid analysis*, and *inductive learning*.

All the these methods are used to identify domain objects and draw relationships among the objects. They can make up for the weaknesses of direct methods when the expert has difficulty expressing his or her knowledge. However, their revealing capabilities are more limited (Olson & Rueter, 1987). They can only illuminate particular aspects of the relationships among the objects in the domain of expertise, but are hard to use to illuminate the inferences the expert is using. In addition, they involve assumptions about the form of the underlying representation, whether it is physical space, lists, networks, or tables, etc. For example, multidimensional scaling assumes that data have come from stored representations of physical n-dimensional space (Olson & Rueter, 1987). Probably their important use in knowledge acquisition is to create data for structured interviews (Greenwell, 1988).

Methods of active knowledge engineer role: As indicated by the method name, a knowledge engineer plays an active role in knowledge acquisition. The following methods are discussed in the literature (Boose, 1989; Greenwell, 1988).

- Participant observation: A knowledge engineer acts an apprentice or otherwise participates in the expert's problem-solving process. This would help the knowledge engineer to better understand the process and the knowledge.
- Teachback interview: A knowledge engineer explains some concepts or reasoning behind a particular process to an expert. This method is suitable to validate and to improve the knowledge engineer's understanding of the elicited knowledge.
- Tutorial interview: An expert delivers a lecture to a knowledge engineer. This method is more suitable at the initial stage of the KA process.

Prototyping methods: Prototyping is an approach for developing a "quick and dirty" version of an ES (i.e., the first working ES. It has also proved to be a powerful tool for elicitation of further knowledge (Greenwell, 1988). Welbank (1983) states that it is useful to get a system working as soon as possible. It keeps the expert interested and acts as focus for eliciting knowledge. When using this method, the expert acts as a user to run a number of examples to test the prototype (i.e., the first working system). During this process, the system weaknesses can be more easily detected and the knowledge base can be extended. Pfeifer (1988) argued that a prototype should be completed as early as possible, since problem areas can frequently be pinpointed at this stage and it is much cheaper to detect or correct problems early in the development life cycle.

Automatic tools: Knowledge acquisition is a very complex and costly cognitive and human interaction process. Besides the problems of the expert's introspection and articulation ability, communication between knowledge engineer and expert might be another problem. Thus, researchers have tried to reduce or eliminate the work load of knowledge engineers. This has resulted in the growth of automatic elicitation techniques. Numerous tools have been developed in recent years (Boose, 1989). They have had great impact on

the "indirect" or analysis component of KA by reducing the tediousness of performing such analysis or by greatly augmenting the type of analysis possible (Benbasat & Dhaliwal, 1989). From the view of knowledge elicitation, at their present level of development, these systems are little more than intelligent word processors with a powerful user interface and graphical capabilities (Greenwell, 1988).

However, some tools do have certain capability to capture expert knowledge to maintain and extend its knowledge bases. For example, TEIRESIAS (Davis, 1976) can support interactive knowledge transfer. If the expert is not satisfied with a diagnosis made by the system, he or she can interact with the system to modify rules or add new rules. TEIRESIAS is able to decide whether a new rule is complete and, if the rule seems not, to pose correct questions to the expert. In this sense, TEIRESIAS can support the transfer of knowledge from an expert to the system and maintain the entire knowledge base complete under the control of the human expert (Pfeifer, 1988). Similarly, other tools (e.g., ID3, ACES; Boose, 1989) based on machine learning can help KA.

Multiple expert KA methods: The knowledge to be acquired may be distributed across a group of experts. Multiple expert KA presents major problems because experts may disagree on the use of concepts and vocabulary, and this disagreement may be tacitly causing confusion. These problems sometimes can be relieved by the appropriate management of the KA process (Greenwell, 1988). The following methods are recommended by some researchers (Greenwell, 1988; Boose 1989; Shaw & Gaines, 1989): *Delphi method, brainstorm* (e.g., Crawford slip method), *consensus decision taking* (e.g., voting, conflict resolution), *normative group techniques* (knowledge engineer leads the expert group to discuss the problem and search for the best solutions).

General/application problem solving methods: Some methods focus on eliciting knowledge for specific tasks (e.g., debugging, diagnosis, etc). Some other methods focus on eliciting knowledge for general tasks (e.g., heuristic classification or heuristic construction) (Boose, 1989; Shaw & Woodward, 1989).

4. DISCUSSION—ARE METHODS DIFFERENT FOR DSS AND ES DEVELOPMENT?

Based on the methods described in Section 3.2, this section tests the presented hypotheses: subsection 4.1 tests Hypothesis I and subsections 4.2 and 4.3 test Hypothesis II.

4.1 *The common methods for DSS and ES development*

As discussed before, in ES development, there should also be a user information requirement stage in addition to the stage of knowledge acquisition from experts. In the user information requirement stage, all traditional information requirement determination methods can be applied to both systems. Besides, all the following methods can be applied to DSS information requirement determination and ES knowledge acquisition with different skills and/or in different contexts. Therefore, Hypothesis I is rejected.

Interviews and questionnaires. Both methods can be applied to knowledge/information acquisition for ES and DSS developments. However, in ES development it is expected that more skills are needed in conducting interviews and building questionnaires (e.g., critical incident questioning strategy; Welbank 1983). This is because the knowledge engineer of an ES would try to understand the expert's mental model in order to have a dense representation of the whole problem-solving process and expertise, and because experts usually have difficulty expressing expertise (Shaw & Gaines, 1989).

Brainstorming. Brainstorming can be applied in DSS development to generate a large number of alternative decisions. It can also be applied in ES development (e.g., Crawford slip method; Boose, 1989).

Group consensus methods (e.g., Delphi method). In DSS development, the Delphi method can be used to obtain "best" judgmental estimate of variables that are difficult or impossible to estimate quantitatively. It can also be implemented as a "model" mechanism to get the group decision in a group DSS. In ES development, the Delphi method can be applied to gather knowledge from multiple experts independently (Boose, 1989).

Deriving information/knowledge from an existing system. In DSS development, if there is an existing system in a similar organization or described in textbooks (handbooks or industry studies), users and analysts can choose that existing system as an anchor and adjust the requirement from it. A similar situation may also occur in ES development.

Evolutive and adaptive design approaches. In DSS development, because users have difficulty pre-specifying their decision support needs without a concrete system to which they can react, and because the decision support needs of users change frequently, the evolutive (i.e., the progressive design of a DSS going through multiple, minimum length of cycles) and adaptive design approaches have been advocated to obtain users' requirements (Davis, 1982; Meador & Rosenfeld, 1986). Similarly, in ES development, a prototype system can act as an aid to further knowledge elicitation (Welbank, 1983), although it has some disadvantages (e.g., scaling-up problems, mixing specification and implementation details) (Twine, 1989). The embedded knowledge acquisition process in Hawkins' negotiation cycle of expert-client interaction (elicitation /modelling /advice /query) can be thought of as an adaptive approach in response to clients who play active roles in this "social process" (Gaines, 1989; Shaw & Gaines, 1989).

Observation and protocol analysis. In ES development, by watching the expert work, the knowledge engineer can discover the objects, relationships, and inferences that the expert is using (Welbank, 1983; Olson & Reuter, 1987). It can also be applied in DSS development to watch the decision maker making decisions. A close cousin of simple observation is protocol analysis. In ES development, the knowledge engineer can ask the expert to "think out loud" while performing the task. As discussed before, in DSS development, verbal protocol analysis can provide knowledge about how to design both generic and domain-specific models. It can be used to determine how users interpret various human-computer interface functions and what makes these functions valuable. It is also useful in evaluating decision-making support, especially for acquiring feedback in an iterative design approach (Todd & Benbasat, 1987).

Methods of active knowledge engineer roles. These methods (e.g., participant observation, teachback interview, and tutorial interview) are useful in ES development (Boose, 1989). In principle, they can also be applied in DSS development, although system analysts might seldom play such active roles, partially because managers in organizations may dislike such methods for political reasons.

Event analysis. In DSS development, events as the identifiable activities in a decision-making process are coded in a map with a time dimension. This can also be used in ES development to record and analyze what the expert does in a map.

Participant analysis. In DSS development, a system analyst needs to identify a number of characteristics of the participants in a decision-making process, such as their organization positions, in order to support their decision-making processes (Bahl & Hunt, 1984). Similarly, in ES development, the knowledge engineer can also identify the characteristics of expert(s) (e.g., the cognitive style) in order to select the best elicitation method or judge which expert has more reliable knowledge.

Decision-making process and content analysis. In DSS development, the decision-making process and content analysis provide a framework to identify various factors that define and influence a manager's behavior through the pre-selection period (e.g., resource, environmental factors), decision period (e.g., urgency, motivations), and post-decision period (e.g., time lag, decision impact). It is a task analysis to find out what kinds of supports a DSS can provide. There is no direct corresponding method for ES development. However, researchers have also advocated some problem-solving ontologies or generic structures approach (Chandrasekaran, 1988; Woodward, 1989; Shaw & Woodward, 1989) to provide a framework to guide the knowledge engineer as to what data to extract, based on the assumption of the existence of applicable generic problem-solving methods.

4.2 *Any methods unique to DSS development?*

Section 3.2.1 surveys some IRE methods for DSS: (1) ROMC; (2) asking; (3) deriving from existing information systems; (4) event analysis; (5) decision participants analy-

sis; (6) decision process and contents analysis; (7) CSF; (8) strategy set transformation; (9) BSP (4 to 9 are under the heading of "synthesis from characteristics of the utilizing systems"); (10) prototyping; and (11) process tracing methods. Asking includes interviews, questionnaires, brainstorming, and group consensus, which can also be applied in KA, as described in Section 4.1. Prototyping is an iterative, evolutive design approach, and can be used in either DSS or ES system development. In addition, Section 4.1 has discussed that both IRE and KA can apply all the above methods except for (7), (8), (9), and (1).

Methods to be used for organizational DSSs. There are some methods primarily for obtaining organization-level DSS requirements (e.g., strategy set transformation, CSF, BSP) (Davis, 1982; Meador & Rosenfeld, 1986). Turban (1993) states that CSF can be used in feasibility studies for both DSSs and ESs. However, since ESs are mainly designed for supporting individuals and groups, in a strict sense, these methods used for information requirement determination in organizational DSS development are not used for ES knowledge acquisition, although they may be thought of as kinds of indirect methods.

ROMC. ROMC is a process-independent method for identifying the user requirement in a DSS (Sprague & Carlson, 1982). As discussed before, in an ES, the representation of the whole problem-solving process is needed. ROMC is too "rough" to be used in ES development.

4.3 *Any methods unique to ES development?*

Section 3.2.2 surveys some KA methods for ES: (1) direct methods; (2) indirect methods; (3) methods of the active knowledge engineer role; (4) prototyping methods; (5) automatic tools; (6) multiple expert KA methods; and (7) general/application problem-solving methods. Direct methods include interviews, questionnaires, observations, and protocol analysis that can be applied to DSS. Multiple expert KA methods contain brainstorming and group consensus methods, which can be used as IRE methods in DSS. General/application problem-solving methods correspond to decision-making process and content analysis in IRE. Prototyping and methods of active knowledge engineer role can be applied to DSS too, as described in Section 4.1. The only ones left are (2) and (5).

Indirect knowledge acquisition methods. These indirect methods make different assumptions about the form of the "underlying representation" and infer what the expert "must have known" from his or her response. They are useful to reveal the complex relationships that experts do not know exactly. The knowledge engineer wants to obtain the high degree of representational homomorphism in terms of structural match and behavioral match between an ES and experts (Benbasat & Dhaliwal, 1989). But this is not the case for DSSs. In a DSS, we only need a sparse representation of the decision process. Although it is still possible to apply these indirect methods to organize data or develop a specific model, it might not have any advantages. However, from another perspective, those methods used in the development of an organizational DSS can be thought of in nature as indirect methods, because they serve to identify commonalities and relationships in information requirements and usage among the decision areas in an organization.

Automatic tools. Automatic tools are highly related to indirect methods and machine learning. In DSS, there are three technology levels—specific DSS, DSS generators, and DSS tools (Sprague & Carlson, 1982; Turban, 1993). DSS generators (e.g., Lotus 1-2-3) and DSS tools (e.g., programming languages) are computer-aided tools to construct a specific DSS. They cannot be used as an IRE method.

Since there are some methods unique to ES or DSS, Hypothesis II is rejected.

5. CONCLUSIONS

From the preceding discussion, it is concluded that almost all knowledge/information acquisition methods can be applied to both DSS and ES development, although they may need different skills or may be applied in different contexts because of involved knowledge sources and types. However, there are still a few methods that may be not applicable to both systems (e.g., methods related with organizational DSSs, ROMC, indirect methods, and automatic tools). Therefore, both extreme Hypotheses I and II are rejected. System

developer in both fields (system analysts in DSS and knowledge engineers in ES) can apply those common methods even though some of the methods originated from the other field. However, they should also be aware that there are some methods unique to the other system, because different system characteristics cause different information/knowledge needs.

DSSs can support individuals, groups, or organizations. Therefore, there are a number of methods particularly to be applied for developing organizational DSSs. Strictly speaking, those methods cannot be applied for ES development because ESs do not support organizations. However, they may be thought of as the IRE methods in DSSs corresponding to indirect knowledge acquisition methods in ESs.

The goal of a DSS is to support the intuition of the decision maker to solve ill specified, complex, *ad hoc*, and unique problems. The computer performs the well understood parts of the problem solving, while users use their goal, intuition, and general knowledge to formulate problems, modify and control the problem-solving strategies, and interpret the results. Therefore, a DSS only has a sparse representation of decision process. Those user requirements can be sufficiently obtained by process-independent methods like ROMC together with other methods. There would be no advantage in applying indirect knowledge acquisition methods to DSS development. On the other hand, the goal of ESs is to replicate and replace experts to solve well specified narrow domain problems. A dense representation is needed and the system will take the initiative in interaction with the user. Thus, the knowledge engineer would need to apply indirect methods or automatic tools to acquire knowledge that even a human expert cannot express explicitly. The "rough" method like ROMC may not be useful in ES development.

Researchers have observed that in certain problem domains both ESs and DSSs may have distinct advantages that, when combined, can yield synergetic results (Turban, 1993). There are a number of ways to integrate ESs into DSSs. One such integrated system is the expert support system (ESS) (Luconi *et al.*, 1986). For those integrated systems, all the methods mentioned in this paper are applicable for their development.

Acknowledgement—The author wishes to thank anonymous reviewers for their helpful comments.

REFERENCES

Bahl, H.C., & Hunt, R.G. (1984). A framework for system analysis for decision support systems. *Information & Management, 7*, 121–131.

Bahl, H.C., & Hunt, R.G. (1985). Problem-solving strategies for DSS Design. *Information & Management, 8*, 81–88.

Benbasat, I., & Dhaliwal, J.S. (1989). A framework for the validation of knowledge acquisition. *Knowledge Acquisition, 1*(2), 215–233.

Boose, J.H. (1989). A survey of knowledge acquisition techniques and tools. *Knowledge Acquisition, 1*(1), 3–37.

Breuker, J., & Wielinga, B. (1987, September). Knowledge acquisition as modelling expertise: The KADS methodology. Paper presented at the 1st European Workshop on Knowledge Acquisition for Knowledge-based Systems, Reading University.

Byrd, T.A., Cossick, K.L., & Zmud, R.W. (1992). A synthesis of research on requirements analysis and knowledge acquisition techniques. *MIS Quarterly, 16*(1), 117–138.

Chandrasekaran, B. (1988). Generic tasks as building blocks for knowledge-based systems: The diagnosis and routine design examples. *The Knowledge Engineering Review, 3*(3), 183–211.

Davis, G.B. (1982). Strategies for information requirements determination. *IBM System Journal, 21*(1), 4–30.

Davis, G.B., & Olson, M.H. (1982). *Management information systems: Conceptual foundations, structure, and development* (2nd edition). New York: McGraw-Hill.

Davis, R. (1976). Applications for meta level knowledge to the construction, maintenance and use of large knowledge bases. Report STAN-CS-76-552. Stanford Artificial Intelligence Laboratory, Stanford University, CA.

Dhaliwal, J.S., & Benbasat, I. (1989, October). A framework for the comparative evaluation of knowledge acquisition tools and techniques. Paper presented at the 4th AAAI-Sponsored Knowledge Acquisition for Knowledge-Based System Workshop, Banff, Canada.

Ericsson, K.A., & Simon, H.A. (1984). *Protocol analysis*. Cambridge, MA: MIT Press.

Gaines, B.R. (1989). Social and cognitive processes in knowledge acquisition. *Knowledge Acquisition, 1*(1), 39–58.

Greenwell, M. (1988). *Knowledge engineering for expert systems*. Chichester: Ellis Horwood Limited.

Keen, P.G.W., & Scott-Morton, M.S. (1978). *Decision support systems: An organizational perspective*. Reading, MA: Addison-Wesley.

Luconi, F., Malone, T.W., & Scott-Morton, M.S. (1986). Expert systems: The next challenge for managers. *Sloan Management Review, 27*(4), 3–14.

Meador, L.C., & Rosenfeld, W.L. (1986). Decision support planning and analysis: The problem of getting large-scale DSS started. *MIS Quarterly, 10*(2), 156–177.

Naumann, J., & Jenkins, M. (1982). Prototyping: The new paradigm for systems development. *MIS Quarterly*, *6*(3), 29–44.

Olson, J.R., & Reuter, H.H. (1987). Extracting expertise from experts: Methods for knowledge acquisition. *Expert Systems*, *4*(3), 152–168.

Pfeifer, R., & Lüthi, H.-J. (1987). Decision support systems and expert systems: A complementary relationship? In H.G. Sol *et al.* (Eds.), *Expert systems and artificial intelligence in decision support systems* (pp. 41–51). Dordrecht: D. Reidel Publishing Company.

Pfeifer, R. (1988). Knowledge acquisition and learning. In S. Savory (Ed.), *Expert systems in the organization*. Chichester: Ellis Horwood Limited.

Shaw, M.L.G., & Gaines, B.R. (1989, July). Knowledge acquisition: Some foundations, manual methods and future trends. Paper presented at the 3rd European Workshop on Knowledge Acquisition for Knowledge Based Systems, Paris.

Shaw, M.L.G., & Woodward, J.B. (1989, October). Mental models in the knowledge acquisition process. Paper presented at the 4th Knowledge Acquisition for Knowledge Based Systems Workshop, Banff, Canada.

Sprague, R.H. (1980). A framework for research on decision support systems. In G. Fick & R.H. Sprague (Eds.), *Decision support systems: Issues and challenges*. Oxford: Pergamon Press.

Sprague, R.H., & Carlson, E.D. (1982). *Building effective DSS*. Engliewood Cliffs, NJ: Prentice-Hall.

Todd, P., & Benbasat, I. (1987). Process tracing methods in decision support systems research: Exploring the black box. *MIS Quarterly*, *11*(4), 493–512.

Turban, E. (1993). Decision support and expert systems: Management support systems (3rd edition). New York: Macmillian.

Twine, S. (1989, July). A model for the knowledge analysis process. Paper presented at the 3rd European Workshop on Knowledge Acquisition for Knowledge Based Systems, Paris.

Welbank, M. (1983). A review of knowledge acquisition techniques for expert systems. British Telecom Research Lab, Working Paper.

Woodward, B. (1989, July). Integrating task demands and problem-solving methods to develop useful taxonomies for knowledge acquisition. Paper presented at the 3rd European Workshop on Knowledge Acquisition for Knowledge Based Systems, Paris.