IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4499

# Automatic Generation of Requirement Scenarios

Gyeung-min J. Kim
School of Business Administration, Portland State University, Post Office Box 751, Portland, Oregon 97207-0751, Ph: (503) 725-3707, Fax: (503) 725-5850, joank@sba.pdx.edu

Rohae Myung
Department of Industrial Engineering, Korea University, 5-1, Anan-Dong, Seongbuk-Gu, Seoul, Korea, Ph: (822)-3290-3392, Fax: (822)-929-5888, rmyung@kuccnx.korea.ac.kr

## ABSTRACT

*Participatory Design (PD) methodology has been used to develop user acceptable systems by accommodating various needs of users at the early stage of system development. However, PD is considered difficult to use because of the lack of tools and techniques. This research developed a theoretical framework for automatic generation of requirement scenarios, each of which describes a possible user requirement for the system. Automatically generated requirement scenarios trigger users to identity missing or unnecessary user requirements and allow users to address what should be and what should not be developed.*

## 1. INTRODUCTION

Failure in identifying user requirements at the early stage of system development leads to resistance from the user group in use of the system. It has been reported that failure in Business Process Re-engineering (BPR) projects is resulted from lack of consideration for human factors issues involved in the introduction of a new technology [Davenport, 1995]. Since BPR is initiated by top-management with assistance from BPR consultants, the requirement of the system is determined based not on user's job satisfaction needs, but on corporate BPR objectives.

Participative Design (PD) methodology has a long history in Scandinavia as an approach to developing user acceptable systems [Carmel et al, 1993]. Regardless of whether the system is an information system or an electronic device, the purpose of PD is to improve usability of the system by accommodating users' perspectives on system uses. Even though PD receives attention from academic circles in Information Systems (IS) area, PD acceptance by North American IS practitioners is relatively low [Carmel et al, 1993]. One of the difficulties in use of PD is attributed to lack of tools and techniques that ease the task of participating users and analysts [Kensing and Munk-Masden, 1993; Greenbaum, 1990].

PD community recognized that identification of use case scenarios describing possible uses of future systems at the front end of the system development is beneficial for developing right system requirements [Greenbaum and Kyng, 1991; Jacobson, 1992; Brown, 1997]. However, brainstorming is a typical methodology for users to create use case scenarios during PD session, which heavily depend on the people skill and experiences of the analysts [Brown, 1997; Ambler, 1995].

According to Nielsen [pp. 88-89, 1993], "It is important to realize that users are not designers, so it is not reasonable to expect them to come up with design ideas from scratch … it is important to realize that participatory design should not just consist of asking users what they want, since users do not know … even what the possibilities are". Nelson indicates the need for tools and techniques to help users identify what they want.

In order to help users identify their user requirements, the objective of this study is to investigate possibilities of generating requirement scenarios automatically. A requirement scenario describes an alternative idea about user requirements. This paper takes the position that the requirement scenarios provide cues to foster user creativity during PD session. Users compare and reason about alternative requirement scenarios, which results in revising/eliminating the alternative user requirements or devising new ones. The revised and devised requirements are included in the final user requirements. During this process, various users' needs and issues are addressed and reflected on the requirement specification. As results, users will be more satisfied with the system; and the system will be more usable.

In order to fulfill the objective of this research, the following research questions are put forth:

1) What are the underlying concepts and techniques of the idea generation tools that can guide the automatic generation of requirement scenarios?
2) Given understanding of idea generation, what are the elements of user requirements that must be represented and manipulated by the tool to generate requirement scenarios?
3) What is the method of representing the elements in order to generate requirement scenarios?

The subsequent two sections investigate the first two questions. Then, the last question will be addressed. Finally, conclusion of this study is presented.

## 2. IDEA GENERATION

Referred to as problem formulation process in decision sciences, understanding problem domain consists of two complementary subprocess of information search and equivocality reduction [Daft and Lengel, 1986; Weick, 1979; Simon, 1977]. Through information search, different viewpoints and issues of the problem are uncovered and broad understanding of the problem domain is generated [Niederman and Desanctis, 1995]. Consensus on the problem is made through equivocality reduction. As results, what should be considered and excluded from the subsequent steps of problem solving are determined [Niederman and Desanctis, 1995; Nutt, 1992].

Compared to heuristics and tools to support problem formulation in general, the tools designed to support information search stage of problem formulation are categorized as idea generation tool. The idea generation tool simulates human's divergent thinking mode, making many connections among problem elements and generating alternative ideas [MacCrimmon and Wagner, 1994; de Bono, 1993; Young, 1991; Ackoff and Vergara, 1981; Guilford, 1967]. In the example of a design of a transportation system [Young, 1991], generic dimensions of the system are first identified: Power Source, Load Container and Control system. Then, possible alternatives for each dimension are identified below.

*Power Source*: Coal-Steam, Electric.
*Load Container*: Conveyor Belt, Flat Car, Enclosed vehicle.
*Control system*: Manual, Automatic.

Then combinations of values of each dimension comprise a total design concept:

1) Coal-Steam/Conveyor Belt/Manual.
2) Coal-Steam/Conveyor Belt/ Automatic.
3) Electric /Conveyor Belt/Manual, etc…

This paper appears in *Challenges of Information Technology Management in the 21st Century,* the proceedings of the Information Resources Management Association International Conference. 

The presentation of the alternatives triggers human's thinking and results in either identification of the new ideas or modification/elimination of the existing ideas in the alternatives.

According to MacCrimmon and Wagner [1994], problems can be analyzed in many different perspectives such as attributes, functions, or purposes-means of the problem. Connections of these problem elements can also come in many different ways. In relational combinations [MacCrimmon and Wagner, 1994, Crovitz, 1970], the problem elements are combined by means of randomly selected relational words such as "above". The purpose of this technique is to show familiar problem elements in a randomly constructed sentence to induce thought about new connections between problem elements.

Given the goal of a system, user requirement determination (RD) for the system is a continuous process of understanding and analyzing the problem domain to determine what should be developed as well as what should not be developed. During this process, the general problems are refined and clarified; the potential problems are uncovered; and the ways to improve or solve the problems are identified. Like problem formulation in decision making, requirement formulation comprises two complementary subprocesses of information search and equivocality reduction. The people involved in RD search for information to determine the problems and look for alternative ways to make their work process better. In this sense, some of concepts and techniques used in idea generation tools to support problem formulation can be applied to the development of idea generation tools to support requirement formulation. In the subsequent section, the following issues will be addressed: (1) how the user requirements are analyzed; (2) what are the elements in user requirements; and (3) how the identified requirement elements are connected to promote idea generation during requirement formulation process.

## 3. REQUIREMENT ELEMENTS AND REQUIREMENT SCENARIOS

User requirement is analyzed in many perspectives, one of which is object-oriented methodology. In object-oriented analysis, requirements are analyzed from perspective of objects such as people or machinery comprising the problem domain. First, individual objects are investigated to find the individual needs or wants from the system (i.e. individual requirements). Then, relationships of individual objects are studied to determine what organizational requirements may be necessary in order to make the system functional in the user's environments. As results, user requirements are formulated in terms of (1) the new services that objects (human actors, organizational units, and system) should provide and (2) the interactions among the objects to achieve the services assigned to the objects. Examples are shown below:

**Examples of User Requirements**

1. Sales Department should be able to tell customer an estimated delivery date with a support from computer.
2. System accesses inventory information by looking up inventory database.
3. System provides inventory information to the sales department.
4. System receives orders through Internet.

In these examples, main objects are: sales department and system as well as order, inventory and customer. "Tell customer an estimated delivery date with a support from computer", "access inventory information by looking up computer", "provide inventory information to the sales department ", and "receives order through Internet" are services of the objects. While the first service is performed by the sales department object, the last three are performed by the system. The combination of an object and its service is considered as Requirement Element (RE). For example, the combination of an object, "sales department" and its service, "tells customer an estimated delivery date with a support from computer" results in a RE, "sales department tells customer an estimated delivery date with a support from computer". REs are building blocks of user requirements. A collection of REs represents a set of user requirements for the system.

The discovery of REs and their relationships, is a difficult task for end-users and heavily dependent upon the experiences and skill of analysts. A supporting tool for this procedure is the one that can generate alternative connections among various REs drawn from similar situations to the user's domain. Theoretically, the alternatives have possibilities of triggering user's idea generation on their own REs and their relationships.

In order to support requirement formulation process, idea generation tool must provide ways to represent and connect the candidate REs to generate alternative ideas about user requirements. For this purpose, a RE is denoted as *object.service* representing that the *object* provides the *service*. The REs are categorized according to the type of service that each RE falls into. Consider the following exemplar REs:

RE1: System.Receive_Orders_Through_Internet.
RE2: Sales.Receive_Orders_From_Floor.
RE3: System.Access_Inventory_Information_By_Looking_Up_Inventory_Database.
RE4: Sales.Access_Inventory_Information_By_Calling_To_Warehouse.

While the first two REs are categorized as the service type, RECEIVE_ORDER, the last two are categorized as the service type, CHECK_INVENTORY. Each RE represents an alternative way of providing a certain type of service by an object.

A series of REs is called as a Requirement Scenario (RS). Examples are shown below:

RS1={RE1, RE3}
RS2={RE1, RE4}
RS3={RE2, RE3}
RS4={RE2, RE4}

Each RE in the RS must have a different service type. In the Requirement Scenario, RS1, while the type of RE1 is RECEIVE_ORDER, the type of RE2 is CHECK_INVENTORY. RS1, describes an alternative set of user requirements: (1) system receives orders through Internet and (2) system accesses inventory information by looking up inventory database. The rules like syntax in linguistics are used to describe valid arrangements of the REs. An exemplar rule is that the services of RECEIVE_ORDER are prerequisite for the services of CHECK_INVENTORY. This rule represents temporal constraints between services.

Each RS is generated automatically and describes an alternative idea about user requirements. The RS provides cues to foster user creativity during participatory design. The RS triggers users to identify missing or unnecessary user requirements and correct misunderstood requirements. By either selecting or modifying one or more RSs, the users formulate their final requirements for the system. As results, users will be more satisfied with the system and the system will be more usable.

The next section discusses the issues such that (1) how an inventory of REs can be obtained; and (2) how the rules of combining the REs are represented to generate RSs.

## 4. AUTOMATIC GENERATION OF REQUIREMENT SCENARIO

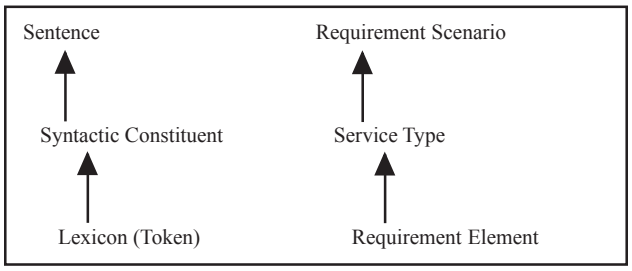
### 4.1 The Source of Requirement Elements

Many companies have similar kinds of business processes, each of which consists of the similar types of services. However, each company implements its services with a different set of means. The first column in table 1 lists the types of services comprising the customer order process. The table indicates that every customer order consists of a generic sequence of services such as RECEIVE_ORDER, CHECK_INVENTORY, and PROCESS_ORDER [Malone, 1993]. For each service type, each of the second and third columns shows a specific means to implement the service type. Case 1 illustrates the process where the service type, RECEIVE_ORDER is accomplished by receive_order_from_Internet. Case 2 describes the process where RECEIVE_ORDER, is implemented by

*Table 1. Requirement Elements (REs) in Customer Order process*
S: Sales department, W: Warehouse, SH: SHipping department

| Service Types | Case 1 | Case 2 | … |
|---|---|---|---|
| RECEIVE ORDER | S.Receive_order_ from_Internet | S.Receive_order_ from_floor | |
| CHECK_ INVENTORY | S.Look_up_ computer | S.Look_up_ computer | |
| PROCESS_ORDER | S.Send_copy_to_ warehouse | S.Put_order_into_ computer | |
| | S.Notify_to_shipping | S.Schedule_delivery | |
| | SH.Schedule_delivery | W.Check_order | |
| | SH.Call_to_customer | SH.Move_product | |
| | SH.Move_product | | |

*Figure 1. Relationships among Grammatical Components*




receive_order_from_floor. The character proceeding a dot (.) operator shows the object who carries out the service: S for Sales Department, SH for Shipping and W for warehouse. The inventory of REs for a service type can be obtained from cases of many companies engaging in the similar process. Given the inventory of the REs for each service type, RSs can be generated.

An RS is a series of REs, describing a sequence of the services performed by objects to complete a work process. For example, the requirement scenario, RS1, in the previous section describes a sequence of services by various objects engaging in customer order process: (1) system receives orders through Internet and (2) system access inventory information by looking up inventory database. In this study, Process Grammar [Pentland, 1995] is used to represent the temporal relationships among REs as linguistic grammar is used to represent the sequential relationships of words. Since the RSs generated based on the grammar, describe a valid work process, the grammar is named as Process Grammar.

### 4.2 Process Grammar

Although the most familiar type of grammar is English grammar, grammar has been used in many areas to describe a set of possible patterns such as circuit patterns in semi-conductor wafer. While linguistic grammar defines a set of valid sentences in a language, circuit grammar defines a set of valid shapes of electronic circuits.

The basic elements of a grammar is called a lexicon [Miclet, 1986] or token. They are treated analytically as the most detailed level of description necessary for the problem at hand. An example of the lexicon is a word of a language. The next level of the grammatical component is called syntactic constituents. In linguistics example, words or phrases are categorized into different syntactic constituents such as noun phrases or verb phrases according to a particular function that each category serves in the syntax of sentence. These constituents can be combined according to grammatical rules to create sentences. Figure 1 shows hierarchical relationships among grammatical components in linguistics.

A grammar provides a framework for generating new instances of a set. In linguistic grammar, a number of valid sentences can be generated from a pool of words. Each sentence is an instance of a certain grammatical rule. For example, from a set of verbs {eats, washes} and a set of nouns {dog, apple}, the following sentences can be generated: "dog eats apple", "dog washes apple", "apple eats dog" and "apple washes dogs". Each sentence is a distinct case of the rule, noun verb noun.

In our study, a Requirement Element (RE) is considered as a token. As words are categorized into noun based on their functionality, REs are categorized according to the type of the services that each RE falls into. Thus, the service types are syntactic constituents in this study. Syntactic constituents have the following characteristics [Pentland,1992]:

1) provide a way of describing the structural features of a pattern without elaborating it all the way down to the specifics of the token.
2) can be nested together.

The service type such as RECEIVE_ORDER meets the first characteristics since it generalizes receive_order_by_mail, receive_order_from_floor and other ways of receiving orders. The service type, PROCESS_ORDER is a nested with another service type, FILL_ORDER_FROM_STORE, FILL_ORDER_FROM_OTHER_STORE, or FILL_ORDER_FROM_ON_ORDER.

The 'service types' are combined according to the various constraints such as temporal and job-control constraints [Pentland, 1995]. For example, the service, CHECK_INVENTORY must be performed later than RECEIVE_ORDER. The constraints govern the way REs are arranged in order to create Requirement Scenarios (RSs). Figure 1 describes relationships among components of the process grammar along with those of the linguistic grammar.

*Figure 2. Grammar for Customer Order Process*

1. *CUSTOMER_ORDER_PROCESS* ∅ RECEIVE_ORDER, CHECK_INVENTORY, PROCESS_ORDER.
2. RECEIVE_ORDER ∅ **S.receive_order_by_mail**.
3. RECEIVE_ORDER ∅ **S.receive_order_from_floor**.
4. CHECK_INVENTORY ∅ **S.look_up_computer**.
5. PROCESS_ORDER ∅ **S.return_order**.
6. PROCESS_ORDER ∅ FILL_ORDER_FROM_STORE.
7. PROCESS_ORDER ∅ FILL_ORDER_FROM_OTHER_STORE.
8. PROCESS_ORDER ∅ FILL_ORDER_FROM_ON_ORDER.
9. FILL_ORDER_FROM_STORE ∅ **S.send_copy_to_warehouse**, **S.notify_to_shipping**, **SH.schedule_delivery**, **SH.call_to_customer,** **SH.move_product**.
10. FILL_ORDER_FROM_STORE ∅ **S.put_order_into_computer**, **S.schedule_delivery,** **W.check_order**, **SH.move_product**.
11. FILL_ORDER_FROM_OTHER_STORE ∅ **S.dial_to_store**, SELECT_PICK_UP_CHOICE.
12. SELECT_PICK_UP_CHOICE -> **S.reserve_stock_for_pick_up**.
13. SELECT_PICK_UP_CHOICE -> **S.transfer_from_stock**.
14. FILL_ORDER_FROM_ON_ORDER ∅ **S.reserve_from_on_order**.

The grammar in Figure 2 describes the grammar for Customer Order Process (COP). Each rule in the grammar has a number for reference. The symbol "->" is read as "consists of." While a bolded element (e.g. **S.receive_order_from_floor)** is a "RE", an element in upper case fonts (e.g. RECEIVE_ORDER) is a "service type". The element in italic font (e.g. *CUSTOMER_ORDER_PROCESS*) represents the domain of the system for which user requirements are developed.

Rule 1 dictates that the Requirement Scenarios (RSs) for *CUSTOMER_ORDER_PROCESS* consist of a series of service types, RECEIVE_ORDER, CHECK_INVENTORY and PROCESS_ORDER. This rule is a generic rule that can be found in any customer order process across types of businesses. The sequence of these service types is determined by temporal constraints on them: CHECK_INVENTORY must be performed after RECEIVE_ORDER and before PROCESS_ORDER.

Although every customer order process has the same types of the services, each implements its services with a different set of services. Rule 2 and 3 illustrate such cases. Rule 2 describe the case where RECEIVE_ORDER is accomplished by the service of the sales department denoted as **S.receive_order_from_mail**.

### 4.3 Generation of Requirement Scenarios.

COP grammar can be used to generate the following alternative Requirement Scenarios (RS). The RSs are grouped according to a common functionality.

*For receive_order_from_floor:*

RS1. return_order
- 1.1. S.receive_order_from_floor, S.look_up_computer, S.return_order.

RS2. FILL_ORDER_FROM_STORE
- RS2.1. S.receive_order_from_floor, S.look_up_computer, S.send_copy_to_warehouse, S.notify_to_shipping, SH.schedule_delivery, SH.call_to_customer, SH.move_product.
- RS2.2 S.receive_order_from_floor, S.look_up_computer, S.put_order_into_computer, S.schedule_delivery, W.check_order, SH.move_product.

RS3. FILL_ORDER_FROM_OTHER_STORE
- RS3.1. S.receive_order_from_floor, S.look_up_computer, S.dial_to_store, S.reserve_stock_for_pick_up.
- RS3.2. S.receive_order_from_floor, S.look_up_computer,

S.dial_to_store, S.transfer_from_stock.
RS4. FILL_ORDER_FROM_ON_ORDER
RS4.1. S.receive_order_from_floor, S.look_up_computer, S.reserve_from_on_order.

These alternative RSs provide Participatory Design (PD) group with insights into new possible alternatives. Besides, each RS reflects various constraints imposed on grammatical rules. The requirement scenarios allow users to identify unnecessary constrains and to modify the existing constraints to accommodate their needs. For example, presentation of RS 2.2 triggers the shipping department to realize that they want the authority of scheduling shipping instead of the delivery schedule being given by the sales department (note that schedule delivery is done by the sales department). The requirement scenario, RS2.2 enables PD group to identify needs of the shipping department for the job autonomy.

The grammar in Figure 2 can be prepared, after system analysts identify minimum user requirements. That is, after the analysts identify that the system domain is the customer order process, the grammar for the customer order process is prepared. The sequence of service types and possible means to implement each service type can be obtained from cases of other companies in the similar situations [Malone et al., 1993].

## 5. CONCLUSION

When novice users are not certain about what to do, they need help. User participation in design requires techniques and tools that enable end users to understand the possibilities for computer support [Grønbaek et al., 1993]. This study demonstrates possibilities of using grammar to generate requirement scenarios that provide PD groups with bases for brainstorming user's various needs. Based on the generated requirement scenarios, users can compare and reason about their requirements; identify missing or unnecessary user requirements; and correct misunderstood requirements. In this way, user oriented issues involved in system requirements can be minimized. In addition, the PD group develops user requirement more efficiently by having a menu of requirement scenarios rather then from scratch.

Grammar has capabilities of representing various constraints. Thus, the grammar generated requirement scenarios can be an efficient medium to communicate constraints of the system to be developed among IS professionals and end-users. The grammar approach is rather a complement for other PD techniques and tools than a substitute. Grammar can complement the weakness of the existing PD techniques, that is qualitative in nature and relies mainly on designer's experience and low technology methods [Carmel et al., 1993]. An automated tool to generate alternative requirement scenarios, which could be a complex sequence of activities, allows PD to become the norm rather that the exception.

## 6. REFERENCE


Abell, P., the Syntax of Social Life: The Theory and Method of Comparative Narratives, New York: Clarendon Press.

Ackoff. R. L. and Vergara, E., "Creativity in Problem Solving and Planning: A Review," European Journal of Operational Research (7:1), 1981, pp. 1-13.

Brown, D., An Introduction to Object-Oriented Analysis, Objects in Plain English, John Wiley and Sons, Inc., 1997.

Carmel, E., Whitaker, R. D., and George J. F., "PD and Joint Application Design: A Transatlantic Comparison", Communications of the ACM, Vol. 36, No.4, 1993, pp. 40-48.

Crovits, H. F., Galton's Walk, Haper & Row, New York, 1970.

Daft R. L. and Lengel, R. H. , "Organizational Information Requirements, Media Richness and Structural Design," Management Science, Vol. 32, No. 5, May 1986, pp. 554-571.

Davenport, T. H., "Will Participative Makeovers of Business Process Succeed Where reengineering Failed?", Planning Review, Vol. 23, No.1, Jan-Feb 1995, pp. 24-29.

De Bono, E., de Bono's Thinking Course, Facts on file, New York, NY, 1993.

Ehan, P., M'lleryd, B. and Sj'gren, D., Playing in reality: A paradigm case. Scandinavian J. Information Systems, 2 (1990), pp. 101-120.

Ehninger D. and Brockreide W., Decision by Debate, 2nd Ed., New York: Harper & Row, 1978.

Goffman, E., Forms of Talk, Philadelphia, PA, University of Pennsylvania Press.

Greenbaum, J., Panel Presentation, Conference on Participatory Design-PDC'90,(Seattle, Wash., 1990), Author's notes.

Greenbaum, J. and Kyng, M., Design at Work: Cooperative Design of Computer Systems, Hillsdale, NJ: Lawrence Erlbaum Associates, 1991.

Guilford, J. P., "The Nature of Human Intelligence", McGraw-Hill, New York, 1967.

Jacobson, I. M., Christerson, M., Jonsson, P., and Övergaard, G., Object-Oriented Software Engineering, New York: Addison-Wesley, 1992.

Kensing, F, and Munk-Masden, A., "PD: Structure in the Toolbox", Communications of the ACM, Vol. 36, No.4, 1993, pp. 78-85.

Malone, T.W., Crowston, K., Lee, J. and Pentland, B., "Tools for Inventing Organizations: Towards a Handbook of Organizational Processes," Proceedings of the Second IEEE Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises, Morgantown, WV, April 20-22, 1993.

MacCrimmon and Wagner, K. R. and Wagner C. "Stimulating Idea through Creativity Software," Management Science, Vol. 40, No. 11, November 1994, pp.1514-1532.

Miclet, L., "Structural Methods in Pattern Recognition, New York: Springer-Verlag, 1986.

Niederman F. and Desanctis, G., "The Impact of a Structured-Argument Approach on Group Problem Formulation, " Decision Sciences, Vol. 26, No. 4, July/August 1995. pp. 451-474.

Nielsen, J., "Usability Engineering", AP Professional, Chestnut Hill, MA, 1993.

Norman, R. J., "Object-Oriented Systems Analysis and Design", Prentice Hall, Upper Saddle River, NJ, 1996.

Nutt, P., "Formulation Tactics and the Success of Organizational Decision Making," Vol. 23, No. 3, May/June 1992. pp. 519-540.

Osborn, A. F., Applied Imagination, Scribner, New York, 1953.

Pentland, B., "Organizing Moves Software Support Hot Lines," Administrative Science Quarterly, 37, 4, pp. 527-548, 1992.

Pentland, B., "Grammatical Models of Organizational Process", Organizational Science, Vol. 6, No. 5, September-October 1995, pp.541-556.

Schank, R. C. and Abelson, R. P., Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures, Hillsdale, NJ: Lawrence Erlbaum, 1977.

Schoemaker, P. H., "Scenario Planning: A tool for Strategic Thinking," Sloan Management Review, 1995 Winter.

Schweitz, R. et al., "Could participative design the answer for us?", Journal for Quality and Participation, January/February, 1997, pp. 34-42.

Simon, H., "The New Science of Management Decisions, Rev. ed., Englewood Cliffs, NJ: Prentice-Hall, 1977.

Smith, G. F., "Defining Managerial Problems: A Framework for Prescriptive Theorizing", Management Science, 1989, 35(8), pp.963-981.

Young, L. F., "Knowledge-Based Systems for Idea Processing Support," Data Base, Vol. 22, No. 1/2, Winter/Spring 1991, pp. 46-50.

Weick, K.E., "The social Psychology of Organizing," Addison-Wesley, Reading, Mass., 1979.