FACETS: A Cognitive Business Intelligence System

The work presented in this paper was partially supported by the Australian Research Council (ARC) under discovery grants DP0880739.

ABSTRACT

A cognitive decision support system called FACETS was developed and evaluated based on the situation retrieval (SR) model. The aim of FACETS is to provide decision makers cognitive decision support in ill-structured decision situations. The design and development of FACETS includes novel concepts, models, algorithms and system architecture, such as ontology and experience representation, situation awareness parsing, data warehouse query construction and guided situation presentation. The experiments showed that FACETS is able to play a significant role in supporting ill-structured decision making through developing and enriching situation awareness.

Keywords: business intelligence, cognitive decision support, decision support systems, situation awareness, mental models

1. Introduction

Business intelligence (BI) is promising to turn ‘data’ into ‘knowledge’ and to help managers survive data tsunami and eventually succeed in decision making. A BI system is capable of providing executives with huge amounts of internal/external business data. However, more data does not equal to more valuable information [1]. Current BI systems can only partially support executives’ management processes [2]. Executives often feel lost when presented with a large body of data concerning decision making. A recent survey, by Economist Intelligence Unit [3], shows 73 per cent of senior managers agreed that it is important to have less but more timely data to improve the quality and speed of decision making. This result corresponds to the research by Sutcliffe and Weber [4] about knowledge accuracy. Their research implies that having a lot of facts about a decision situation is less important than having a clear and consistent overview picture.

Decision support systems (DSS) are envisioned as “executive mind-support systems” that are expected to support decision-making from human cognitive perspectives [5]. Nevertheless, BI systems are essentially data-driven decision support systems. OLAP-based ad hoc query and reporting are mainly pre-defined information representation. During a decision process, managers are provided with information in the form of reports, ad hoc analysis, or some so called knowledge, which is pulled out of a data warehouse according to pre-defined queries, such as SQL sentences and multidimensional expressions (MDX). The emphasis is manipulation of large volumes of business data, rather than supporting managers’ decision making from a cognitive perspective.

The focus of this study is to investigate how various cognitive models can be incorporated into

traditional information systems. A prototype DSS system was designed, developed and evaluated based on the Situation Retrieval (SR) model.

The remaining of this paper is organized as follows. After the introduction, the related works are reviewed in Section 2. Section 3 presents the system architecture of FACETS. The ontology design is described in Section 4. Section 5 introduces the mental model, experience and cue maps. Section 6 details the situation awareness parsing procedure. The process of query construction is described in Section 7. Section 8 shows the process of situation presentation. In Section 9, experiments are done to evaluate the FACETS. Conclusions are given in Section 10.

2. Related Work

Managers' cognition plays an important role in business decision making, as has been noted by many researchers. In behavioral organization theory, managers' cognition acts as a filter between inter-organizational and intra-organizational environments, which helps managers to search for selective information concerning functions of the business and certain organizational actions [6]. Similarly, based on a survey of 12 Fortune 500 companies, Donaldson & Lorsch [7] concluded that senior executives simplify business reality by employing interrelated beliefs to filter irrelevant information. The simplified business environment helps executives to gain better understanding of their business during strategic decision making, as concluded by Porac & Thomas [8]. Mintzberg [9] categorized managers' work into ten different roles and connected them with managers' mental models. He found that managers spend most of their time communicating with other people and thinking, by which a series of mental models are built. Isenberg [10] observed that higher level decision making is mainly based on managers' intuition rather than 'choosing' the best one from a number of identified alternatives. Managers are skillful at using historical experience to envision future scenarios of the company, by which they predict potential threats and possible opportunities. In dynamic, ill-structured environments, managers have little time to conduct thorough rational reasoning. Ironically, managers tend to quickly assess a decision situation by comparing a current decision problem with their past decision scenarios [11].

Managers' cognitive abilities are nevertheless subject to many cognitive biases. A cognitive bias is a distortion pattern in human mind which leads to a perception, judgment, or reliability that deviates from the reality [12]. Cognitive biases might be useful in certain circumstances, but they are more likely to cause serious mistakes in decision making. For example, people tend to accept new information that confirms their preconceptions and to avoid conflicting ones. Russo & Schoemaker [13] described ten most common mistakes in decision making related to cognitive biases: plunging in, frame blindness, lack of frame control, overconfidence in your judgment, shortsighted shortcuts, shooting from the hip, group failure, fooling yourself about feedback, not keeping track and failure to audit your decision process. Most cognitive biases are hard to avoid and they are attributed to different psychological biases. For example, judgmental biases are caused by judgmental rules and heuristics employed by people to reduce difficult mental tasks to a simpler one [14]. However, well-designed information systems are helpful for manger to overcome some negative cognitive biases [1, 15].

Cognitive map, as the knowledge representation technique of human mental models, has received wide research attention in DSS community. A number of DSS have been developed to support manipulation of cognitive maps.

An early DSS called SPRINT (Strategic Plan and Resource INTegration), was developed by Carlson and Ram's [16]. SPRINT is an executive planning support system which can be used by managers to explicitly represent planning models which are of implicit nature in managers' minds. The visual representation of planning models is based on manages' mental models. The concept nodes and links between concepts can be created by managers according to their understanding, thoughts about the company. SPRINT also supports heuristic rules and goal-oriented communication between different managers. The cognitive aspects of SPRINT lie in that it supports visual representation and dynamic creation of managers' mental models regarding business plan formulation. Although managers' cognition is supported in terms of information systems in a limited degree, SPRINT represents one of the early research efforts toward cognitive decision support in DSS community.

A conceptual DSS *Cognitive Lens Support System* was described by Yadav and Khazanchi [15]. They proposed the concept *cognitive lens* as the description of mental models from information systems perspective: 'A cognitive lens converts filtered information into a set of constructs and their interrelationships of the real world'. The description of the cognitive lens support system revolves around the inquiry of cognitive lenses stored in a database. They proposed three categories of IS functions for inquiry of cognitive lenses: introspective, dialectic, and eclectic, which allow managers to examine their past experience for a specific decision problem, to compare their own experience with others, and to aggregate multiple pieces of experience. The major argument relying on the cognitive lens support system is that IS support provided to managers through 'cognitive orientations' will facilitate better understanding of ill-structured problems. Compared to previous research, the cognitive lens support system illustrates a more comprehensive analysis about the significance and IS techniques of supporting managers' thinking process for business decision making, although at a conceptual level lacking empirical validation.

Following the cognitive lens support system [15], Chen and Lee [5] developed a cognitive DSS for strategic decision making. Similarly, their system also includes three supporting modules: retrospective, introspective, and prospective. The first module provides the manager with tools to manage business cases, experience, other people's views, speculations, and even rumors. The second module is used to explore and represent the manager's own mental models. And the last module provides the user aids in forward thinking by creating and managing future business scenarios. An exploratory assessment was conducted to evaluate this system by interviewing real business executives from three different industrial sectors. The evaluation results show that cognitive decision support can be delivered by aiding managers explicitly representing, exploring their mental models.

Cognitive maps, particularly fuzzy cognitive maps, are also employed as a reasoning mechanism for developing domain-specific DSS. Lee and Kim [17] developed a bidirectional (downward or upward) inference system based on fuzzy cognitive maps to solve highly unstructured problems in stock investment domain. Noh and co-researchers [18] combined cognitive map technique with case-based reasoning to solve credit analysis problems. Konar and Chakraborty [19] proposed a unsupervised learning and reasoning model based on fuzzy cognitive maps which is implemented with Petri nets. More recently, Stylios and co-researchers [20] used fuzzy cognitive map technique to assist medical professionals in crucial clinical judgments.

Recognizing the implications of managerial intuition to handling dynamic, ill-structured business problems, Kuo [21] proposed an ecological cognitive model of managerial intuition for executive support system (ESS) development. According to this model, managers are capable of intuitively assess the situation through perception-action cycles during sensorimotor (perception and actions) is

combined with the memory processors (mental models). Cognitive decision support of ESS is then emphasized by modeling the ecology of managers: the interplay between human and environment. The claimed contribution is using his model to guide the development of practical ESS. The impact of computerized cognitive aids was tested in the context of strategy execution process by Singh [22]. This research identified two sorts of specific cognitive requirements in strategy execution processes: memory support and strategy support. The former is intended for compensation for managers' limited attentional resources. The latter is for managers' monitoring abilities. Positive and significant relationships were found between the efficiency and effectiveness of the strategy execution process and computerized cognitive aids.

Aligning with "executive mind support systems", the Situation Retrieval (SR) model was proposed in our earlier work [23].

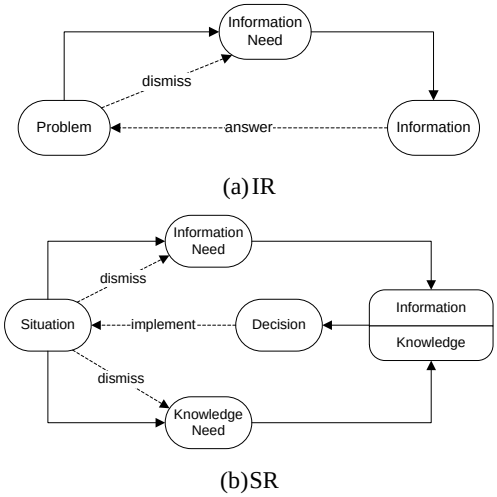


Figure 1 Information Retrieval and Situation Retrieval

We proposed the situation retrieval model based on the information retrieval (IR) model. IR systems are concerned with representing, storing and finding information desired by users [24]. Traditional IR systems are based on the best-match principle: the matching between documents and statements of the queries put forward by the user [25]. IR systems cater for the users' information needs through searching, locating and obtaining target information. A simplified IR model is shown in Figure 1 (a) [24, 25]. With a problem, the user of an IR system will raise a need for relevant information which can be used to solve the problem. The information need motivates the user to interact with the IR system and search for the desired information. Once the user acquires the desired information, the user's problem is solved and the user's information need is dismissed.

We define situation retrieval as the process of searching for situation information as well as situation knowledge for the purpose of decision making. Figure 1 (b) illustrates the situation retrieval model. A situation retrieval process begins with a decision situation. The decision situation leads to the decision maker's information need and knowledge need. Respectively, the information need and knowledge need motive the decision maker to seek information (situation information) and knowledge (situation knowledge) relevant to the decision situation for the purpose of decision making. Once a satisfying decision is made and implemented in the decision situation, the corresponding information need and

knowledge need are dismissed.
The contribution of the research is to design and develop a software system called FACETS based on the SR model. FACETS allows a manager to describe his/her SA in the form of English. Based on the domain knowledge, FACETS parses the manager's SA and constructs data warehouse queries. Queries are submitted to the data warehouse for retrieving relevant situation information. The retrieved situation information is presented to managers according to the navigation knowledge extracted from the manager's experience. The goal of FACETS is to assist managers to develop and enrich their SA for decision making, and to provide decision makers cognitive decision support in ill-structured decision situations.

3. FACETS System Architecture

FACETS allows a manager to describe his/her situation awareness (SA) in the form of English. Based on the domain knowledge, FACETS parses the manager's SA and constructs data warehouse queries. Queries are submitted to the data warehouse for retrieving relevant situation information. The retrieved situation information is presented according to the navigation knowledge extracted from the manager's experience. The goal of FACETS is to assist managers to develop and enrich their SA for decision making.
The prototype system FACETS is comprised of eight subsystems: *Data Warehouse System, Ontology Management, Experience Management, Situation Awareness Management, Situation Awareness Parsing, Situation Awareness Annotating, Query Builder* and *Situation Presentation.* Each subsystem includes a number of components which are shown in Figure 2.
FACETS was built on the platform of a data warehouse, which includes a central data warehouse and other components for extracting data from business operational systems, managing meta data and analyzing data.
The functions of FACETS are based on an ontology and experiences in the knowledge base. Users (e.g., business managers or IT engineers) use the *Ontology Management* subsystem to create and edit the ontology. The data of the ontology is stored in the *Ontology Base*. The *Ontology Management* subsystem also provides a set of functions for operating the ontology, such as traversal, editing and computing class similarities. The *Experience Management* subsystem provides functions for creating, storing, and editing managers' experiences, and extracting navigation knowledge for the *Situation Presentation* subsystem.

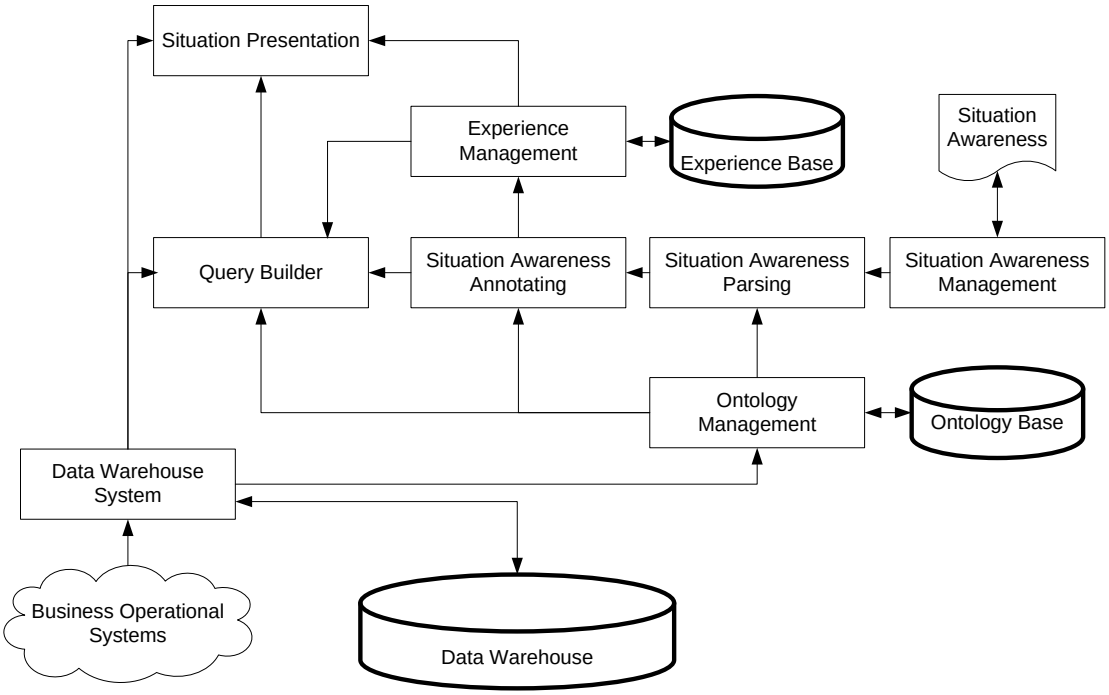


Figure 2 The Architecture of FACETS

The *Situation Awareness Management* subsystem is the interface via which managers input their SA data. The SA data is stored in the form of persistent hard disk files and then sent to the *Situation Awareness Parsing* subsystem for extracting SA tokens. SA tokens are the inputs of the *Situation Awareness Annotating* subsystem responsible for generating SA triples. SA triples are the basis of query construction which is done by the *Query Builder* subsystem. The processes of parsing, annotating SA, and constructing queries are based on the domain ontology. Thus, *Situation Awareness Parsing*, *Situation Awareness Annotating* and *Query Builder* have inputs from the *Ontology Management* subsystem. In addition, constructing queries also relies on experience which comes from the *Experience Management* subsystem. The constructed queries are executed in the *Situation Presentation* subsystem for the retrieval of situation information. The retrieved situation information is finally presented to managers.

4. Ontology Design

In computing science, an ontology is a specification of a data model about classes (concepts) and their relationships [26]. In an ontology, different relationships exist between classes. Subsumption is a very basic type of class relationship. An example of a subsumption relationship is the one between PRODUCT[1] and NOTEBOOK. NOTEBOOK belongs to PRODUCT: any individual (instance) of NOTEBOOK is also a member of PRODUCT. In a specific domain, additional types of relationship need be defined in order to meet different semantic requirements. For instance, there is a *wrote*

[1] In this paper, ontology classes (concepts) are presented in full upper-case (e.g., MARKETING), while the properties of a class are only capitalized (e.g., Sales Amount).

relationship between AUTHOR and BOOK, and an *is-topped-with* relationship between PIZZA and TOPPING. It is noticeable that a hierarchical taxonomy should be defined if only subsumption relationships between ontology classes are depicted. This section first defines a new type of relationship called ***property-share relationships*** for knowledge needs construction. It then proposes some new concepts including class tree, class graph, and class similarity.

4.1 Property-Share Relationships

In real life, people learn of things by obtaining information about them. The information can be either direct or indirect, subject to the observation method. Observing a thing directly, people can obtain direct information. Indirect information can be obtained from intermediaries. For example, people can image how a child (the target) looks like according to his/her parents' (the intermediary) appearance. Indirect information can be considered as a sort of relationship between the intermediary and the target. The information conveyed by this relationship is actually based on the properties of the intermediary which are shared with the target. This information is the intermediary's contribution to the understanding of the target. In other words, we can know something about the target by examining the shared properties of its intermediaries. For the previous example, the child shares some properties (facial appearance) with his/her father. We refer to this kind of relationships between ontology classes as ***property-share relationships***.

A property-share relationship between two classes is a relationship directed from the intermediary class to the target class. Let $\boldsymbol{A}=(\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots \boldsymbol{a}_m)$ and $\boldsymbol{B}=(\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots \boldsymbol{b}_n)$ be an intermediary class and a target class respectively, $\boldsymbol{a}_i$ be the ith property of class A, and $\boldsymbol{b}_j$ be the jth property of class $\boldsymbol{B}$. The property-share relationship from $\boldsymbol{A}$ to $\boldsymbol{B}$ is defined as follows:

Definition 1 Property-Share Relationship (r^s)

$$r^s(\boldsymbol{A}, \boldsymbol{B}) := \{ <\boldsymbol{a}_i, \boldsymbol{b}_j, \boldsymbol{f}_i> \mid \boldsymbol{a}_i \in \boldsymbol{A}, \boldsymbol{b}_j \in \boldsymbol{B} \text{ and } \boldsymbol{f}_i(\boldsymbol{a}_i) = \boldsymbol{b}_j \},\ i=1,\ldots,m,\ j=1,\ldots,n.$$

In this definition, $\boldsymbol{f}_i$ is a ***property transformation function*** which maps a property ($\boldsymbol{a}_i$) of $\boldsymbol{A}$ to a property ($\boldsymbol{b}_j$) of $\boldsymbol{B}$. $\boldsymbol{f}_i$ is defined according to specific application requirements. In most cases, $\boldsymbol{f}(\boldsymbol{p}) = \boldsymbol{p}$ can be adopted, i.e., the target class and the intermediary class share properties directly without any transformation. Let us look at an example to illustrate different property transformation functions.

Intermediary class: PRODUCT RELEASE = (Release Date, Product ID)

Target class: SALES = (Start Date, End Date, Product Model)

r^s (PRODUCT RELEASE, SALES) = {<Release Date, Start Date, $\boldsymbol{f}_1$>, < Product ID, Product Model, $\boldsymbol{f}_2$>}, where $\boldsymbol{f}_1(\boldsymbol{p}) = \boldsymbol{p} + \boldsymbol{N}$, N is a period of time, and $\boldsymbol{f}_2(\boldsymbol{p}) = \boldsymbol{p}$.

With the property-share relationship $\boldsymbol{r}^s$, the Start Date of SALES can be obtained from the Release Date of PRODUCT RELEASE by applying property transformation function $\boldsymbol{f}_1$. According to $\boldsymbol{f}_1(\boldsymbol{p}) = \boldsymbol{p} + \boldsymbol{N}$, Start Date = Release Date + **N**. Suppose **N** = 5 days. Given the date when a product is released, a reasonable start date of this product's sales can be determined. The Product Model of SALES can be obtained from Product ID of PRODUCT RELEASE by property transformation function $\boldsymbol{f}_2$. Since $\boldsymbol{f}_2(\boldsymbol{p})$

= $\boldsymbol{p}$, Product Model = Product ID. Thus, $\boldsymbol{r^s}$ enables us to gain insights into the sales of a product through available data about its release, even though we do not have direct sales data at hand. Well-defined property-share relationships have broad implications in business management, as they provide us with a means to perceive useful information indirectly.

With reference to the definitions of ontology relationships, we have also defined two data structures to represent a domain ontology: ***class tree*** and ***class graph***.

4.2 Class Tree

The building blocks of class trees are nodes and links. A node of a class tree denotes a class in the ontology. The subsumption relationship between two ontology classes is represented as a link in the class tree. Figure 3 is an example of class tree. Of the property set of a class, $\boldsymbol{A=(a_1, a_2, \ldots a_m)}$, we define a special one as ***taxonomy property***, which is used to distinguish its child classes. For example, PRODUCT has four children: BIKE, COMPONENT, CLOTHING and ACCESSORY. These four child classes are differentiated from each other by the Product Category property. Therefore, Product Category is the taxonomy property of class PRODUCT. Although a child class inherits all properties from its parent class, still the child class might have a different taxonomy property to its parent class.

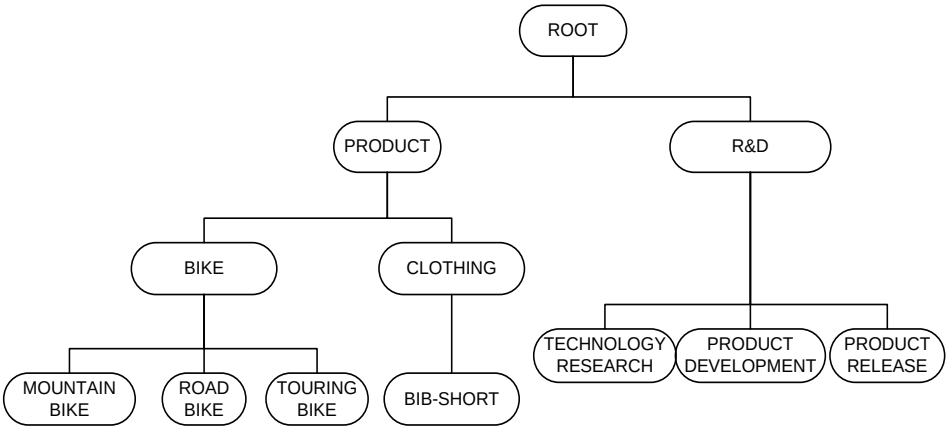


Figure 3 An Example of Class Tree

A class can be one of three class types: ***general class***, ***abstract class*** and ***meta class***. In the hierarchy of a class tree, general classes are the lowest level classes (leaf nodes), and can have specific instances. Compared to general classes, abstract classes are higher level classes in a class tree and do not have direct instances. Therefore, abstract classes cannot be identified by instances. We use semantic parsing to deduce abstract classes from managers' SA descriptions. Meta classes are auxiliary classes which are used to characterize general classes and abstract classes. A meta class has only one property: value. A meta class is usually a property of a general or abstract class. For example, MONEY may correspond to Sales Amount which is a property of SALES.

4.3 Class Graph

A ***class graph*** is another part of the representation of an ontology in addition to its class tree. Compared

to the subsumption relationships in the class tree, property-share relationships are represented in the class graph. The class graph is a directed graph representing the same set of classes as the class tree. Each vertex denotes a class; each directed edge denotes a property-share relationship.

There are two types of edges in a class graph: monodirectional lines and bidirectional lines. A monodirectional edge denotes a property-share relationship, e.g., the edge from PRODUCT SUBCATEGORY to PRODUCT denoting r^s(PRODUCT SUBCATEGORY, PRODUCT). A bidirectional edge denotes two property-share relationships, e.g., the edge between SALES and CUSTOMER denoting r^s(SALES, CUSTOMER) and r^s(CUSTOMER, SALES). As shown in Figure 4, most property-share relationships are symmetric, i.e., both r^s(A, B) and r^s(B, A) are valid property-share relationships.

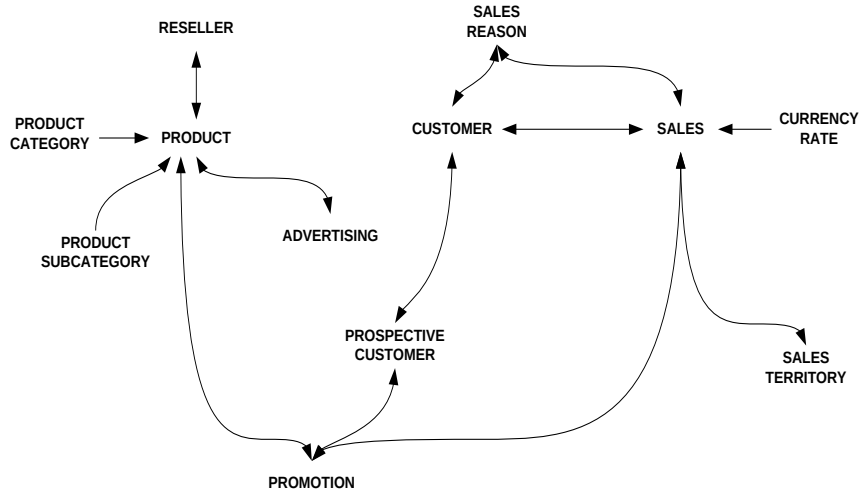


Figure 4 An Example of Class Graph

4.4 Class Similarity

- ***Class similarity***

It is used to retrieve an alternative class when the required class cannot meet the specified criteria. In an ontology, the similarity between two classes is computed based on the class tree and the class graph of this ontology. First, their similarity is evaluated in the class tree (***tree similarities***) and class graph (***graph similarities***) individually and then aggregated together to form the class similarity.

- ***Tree Similarity***

The class tree of an ontology is a hierarchical structure. The methods for computing class similarities within a hierarchical ontology can be grouped into two basic categories: the edge based methods and the node based methods [27]. Schickel-Zuber & Faltings compared various existing methods and concluded that there are only marginal differences between different methods in terms of mean absolute error measure. This conclusion encourages us to adopt a relatively simple similarity evaluation method as shown in the following equation [28].

Equation 1 Tree Similarity (*ts*)

$$ts(a,b) = 1 - \left(\frac{len(a,b)}{2D} \right)$$

where, ***a*** and ***b*** are two nodes of a class tree and ***len***(***a***, ***b***) is the length of the path from node ***a*** to node ***b*** in the class tree. The length of a path in the class tree is defined as the number of nodes in the path. *D* is the maximum depth of the class tree.

- ***Graph Similarity***

A class graph represents the property-share relationships between ontology classes. Let ***a***, ***b*** be two classes in a class graph and $r^s(a, b)$ be the property-share relationship from ***a*** to ***b***. The properties of ***a*** defined in $r^s(a, b)$ can be transformed into corresponding properties of ***b*** by a transformation function. The more properties of ***a*** are transformed to ***b***, the stronger $r^s(a, b)$ is. Therefore, the property-share relationships, to some extent, reflect the degree to which classes are related to each other: the stronger a property-share relationship between two classes, the closer the two classes. We defined a new graph similarity using the equation below.

Equation 2 Graph Similarity (g*s*)

$$gs(a,b) = \prod_{edge(c_i,c_j) \in path(a,b)} \left(\frac{|\mathrm{r}^s(c_i,c_j)| + |\mathrm{r}^s(c_j,c_i)|}{2|c_i|} \right).$$

In this equation, ***a***, ***b*** are nodes, ***path***(***a***, ***b***) is the shortest path from vertex ***a*** to ***b***, ***edge***(c_i, c_j) is one of the series of edges in ***path***(***a***, ***b***), $r^s(c_i, c_j)$ and $r^s(c_j, c_i)$ are the property-share relationships from c_i, to c_j and from c_j to c_i, respectively, $|\mathrm{r}^s(c_i,c_j)|$ is the number of properties of c_i that can be transformed into corresponding properties of c_j, and $|c_i|$ is the number of properties of c_i.

To combine the tree similarity and the graph similarity, the class similarity between two classes can be computed using the following equation.

Equation 3 Class Similarity (*sim*)

$$sim(a,b) = \mu \cdot ts(a,b) + (1-\mu) \cdot ts(a,a^s) \cdot ts(b,b^s) \cdot gs(a^s,b^s).$$

The class similarity of two classes is calculated as the weighted sum of tree similarity and graph similarity. In this equation, $\boldsymbol{\mu}$ ($\mathbf{0} \le \boldsymbol{\mu} \le \mathbf{1}$) is the weight of the tree similarity which is determined by experts according to specific applications. a^s, b^s are respectively the ***surrogate classes*** of ***a*** and ***b***. In a class graph, the surrogate class of a class ***c*** is the one that has the greatest tree similarity with ***c*** compared to all other classes of the class graph. The class graph of an ontology only includes part of classes defined in the class tree. Those classes not involving property-share relationships will not appear in the class graph, e.g., most meta classes. For each of these classes, a surrogate class is used to compute the graph similarities with other classes, and the tree similarity between the surrogate class and the original class is used to adjust the graph similarity. Please note, if class c is also contained in the class graph, c^s

= c and then $ts(c, c^s) = 1$.

5. Experience Representation and Cue Maps

This section defines the concept of experience, the related basic concept for experience representation, the concept of cues and cue maps.

5.1 Experience Representation

Mental models can be elicited using the cognitive mapping technique [29-31]. We refer to computerized mental models as ***experience***. Based on related work on cognitive mapping, we defined experience and related basic concepts for experience representation.

Definition 2 Causal Relationship (r^c)

Let $C = \{c_1, c_2, \ldots, c_n\}$ be a concept set. The causal relationship from c_i to c_j $\boldsymbol{(i, j \leq n)}$ is defined as follows:
$\boldsymbol{r^c}$ $\boldsymbol{(c_i, c_j)}$:= $\boldsymbol{(c_i, c_j)}$, where $\boldsymbol{c_i, c_j \in C, c_i \neq c_j}$, and $\boldsymbol{c_i}$ is a cause of $\boldsymbol{c_j}$.

A causal relationship defines an ordered pair of concepts, of which the former concept ($\boldsymbol{c_i}$) is a cause of the latter concept ($\boldsymbol{c_j}$). We refer to $\boldsymbol{c_i}$ as a ***cause concept*** of $\boldsymbol{c_j}$ and accordingly $\boldsymbol{c_j}$ as an ***effect concept*** of $\boldsymbol{c_i}$. In a causal relationship, the cause concept is a ***direct*** cause of the effect concept. The cause concept directly affects the effect concept in some ways without any intermediaries. ***Indirect*** causes do not guarantee causal relationships between concepts. For instance, if we have $\boldsymbol{r^c}$ $\boldsymbol{(a, b)}$ and $\boldsymbol{r^c}$ $\boldsymbol{(b, c)}$, we can conclude that $\boldsymbol{a}$ is a cause of $\boldsymbol{c}$. However $\boldsymbol{r^c}$ $\boldsymbol{(a, c)}$ does not necessarily exist. In this case, $\boldsymbol{a}$ is an indirect cause of $\boldsymbol{c}$. We refer to indirect causal relationships as ***mediated causal relationships***.

Definition 3 Mediated Causal Relationship (r^m)

Let $\boldsymbol{C}$ be a concept set. The mediated causal relationship is as follows:
$\boldsymbol{r^m(a, b)}$:= $\boldsymbol{(a, b)}$,

where, $\boldsymbol{a \in C, b \in C, a \neq b}$,

and, $\boldsymbol{\exists C' \subseteq C, C' = (x_0, x_1, x_2, \ldots, x_p, x_{p+1}), p \geq 1, x_0 = a, x_{p+1} = b,}$

such that $\boldsymbol{\forall x_i \in C', 0 \leq i \leq p, r^c(x_i, x_{i+1})}$ is a causal relationship.

According to this definition, a mediated causal relationship from one concept to another concept implicitly defines an ordered tuple of concepts between the two concepts. A valid causal relationship exists for each pair of adjacent concepts.
Based on Definition 2, an experience can be defined as follows.

Definition 4 Experience E

$\boldsymbol{E := (C, R)}$,
$\boldsymbol{C = \{ c_1, c_2, \ldots, c_n \}}$, $\boldsymbol{c_i}$ is a concept of concern ,

$$\boldsymbol{R} = \{\, r^c(c_i, c_j) \mid c_i \neq c_j,\ c_i \in \boldsymbol{C},\ c_j \in \boldsymbol{C} \,\},$$

where, $r^c(c_i, c_j)$ is a causal relationship directed from c_i to c_j.

According to Definition 4, an experience has two parts: a set of concepts and a set of causal relationships between concepts. A concept represents an entity or object of an individual's concern. Examples of concept in the business domain are SALES, PRODUCT RESEARCH, MARKETING, and SHARE PRICE. The causal relationship between two concepts in an experience is required to be direct. For example, PRODUCT RESEARCH directly affects SALES; SHARE PRICE directly affects MARKETING.

The representation of an experience is a directed graph which we call an ***experience map***. In an experience map, concepts are represented as vertexes and causal relationships are represented as directed edges between vertexes. Figure 5 is an example of an experience map. From this experience map, we can see how different factors affect each other. For instance, INTERNET SALES is affected by PROMOTION, PRESALE SERVICE, DELIVERY, and ADVERTISING. ADVERTISING is also affected by another three factors: TV, RADIO, and NEWSPAPER.

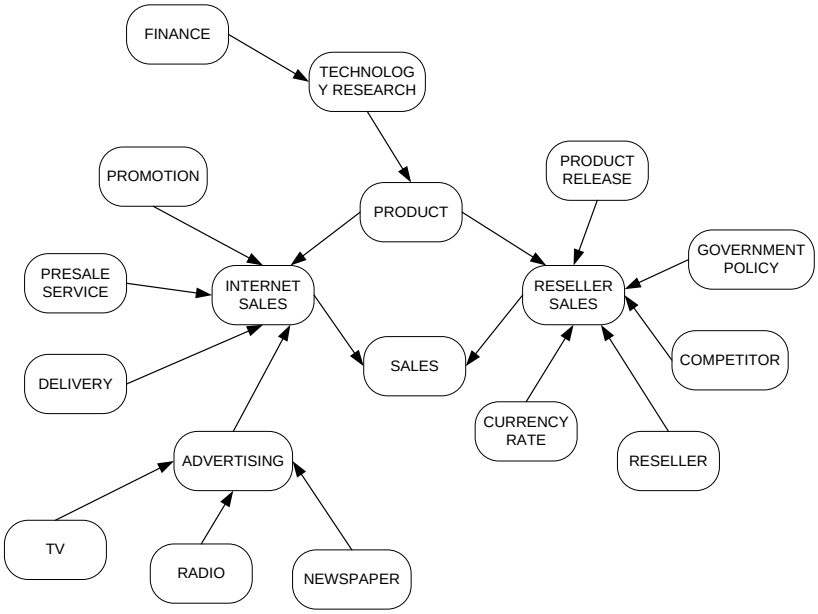


Figure 5 An Experience Map Example

5.2 Cues

Definition 5 Cue ($\check{\boldsymbol{E}}$)

Let $\boldsymbol{E}(\boldsymbol{C}, \boldsymbol{R})$ be an experience and $\boldsymbol{c}$ be a concept of $\boldsymbol{E}$ ($\boldsymbol{c} \in \boldsymbol{C}$). We define the cue $\check{\boldsymbol{E}}$ of $\boldsymbol{c}$ in $\boldsymbol{E}$ as follows.

$$\check{\boldsymbol{E}} := (\check{\boldsymbol{C}}, \check{\boldsymbol{R}}),$$

where $\check{\boldsymbol{C}}$ *is a concept set,* $\check{\boldsymbol{R}}$ *is a causal relationship set,*

$$\forall c_i \in C,\ c_i \begin{cases} \in \check{\boldsymbol{C}}, \text{ if } r^c(c_i,c) \text{ or } r^m(c_i,c) \text{ exists} \\ \notin \check{\boldsymbol{C}} \text{ otherwise} \end{cases}$$

and $\check{R}$ **:=** $\{\, r^c(c_i, c_j) \mid r^c \in R,\ c_i \neq c_j,\ c_i \in \check{C},\ c_j \in \check{C} \,\}$**.**

According to Definition 5, the cue $\check{E}$ of a concept c in an experience E has two parts: a set of concepts ($\check{C}$) and a set of causal relationships ($\check{R}$) between concepts. The concept c per se is an element of C. Furthermore, all other related concepts, from each of which there is a direct or mediated causal relationship to $c \in C$. In other words, given a specific experience, the cue of a concept includes all factors in an experience which affect this concept no matter whether it is direct or indirect.

The representation of the cue of a concept is also a directed graph, which we call ***cue maps***. In a cue map, a concept is represented as a vertex and a causal relationship is represented as a directed edge from one vertex to another. In a decision situation, a concept represents a problem or issue, e.g., sales. The corresponding cue of the concept implies possible factors, information or solutions to the problem.

An abstract experience is shown in Figure 6. Let this experience be $E := (C, R)$. In experience E, all concepts, except for d, are cause concepts; Concepts c, f, and d are effect concepts.

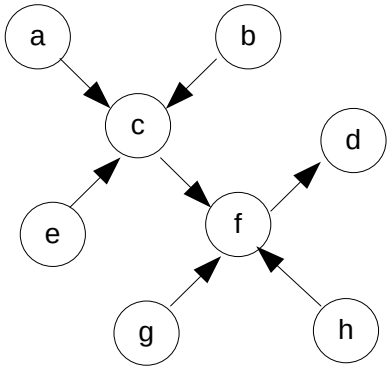


Figure 6 An Abstract Experience Map

The concept set is $C = \{a, b, c, d, e, f, g, h\}$.

The causal relationship set is $R = \{r^c(a, c), r^c(b, c), r^c(e, c), r^c(c, f), r^c(g, f), r^c(h, f), r^c(f, d)\}$.

All mediated causal relationships are $r^m(a, d), r^m(b, d), r^m(c, d), r^m(e, d), r^m(a, f), r^m(b, f)$ and $r^m(e, f)$.

According to Definition 5, the cues of different concepts can be extracted as follows:

$\check{E}(a) = (\{a\}, \varnothing)$

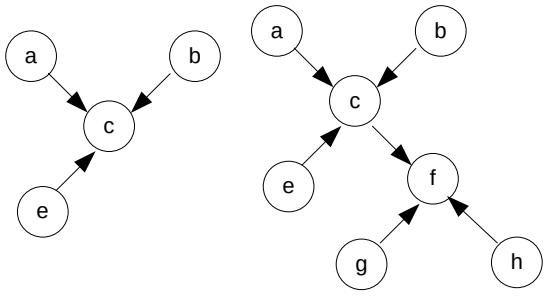
$\check{E}(b) = (\{b\}, \varnothing)$

$\check{E}(c) = (\{a, b, e\}, \{r^c(a, c), r^c(b, c), r^c(e, c)\})$

$\check{E}(f) = (\{a, b, c, e, f, g, h\}, \{r^c(a, c), r^c(b, c), r^c(e, c), r^c(c, f), r^c(g, f), r^c(h, f)\})$

$\check{E}(d) = (\{a, b, c, d, e, f, g, h\}, \{r^c(a, c), r^c(b, c), r^c(e, c), r^c(c, f), r^c(g, f), r^c(h, f), r^c(f, d)\})$

Cue maps $\check{E}(c)$ and $\check{E}(f)$ are shown in Figure 7 (i) and (ii) respectively.



(i) (ii)

Figure 7 Cue Maps

Given managers' SA, concepts are extracted. The cue maps of all concepts extracted from managers' SA are related cue maps. Cues are fused into navigation knowledge through merging all the related cue maps. Navigation knowledge is defined as a special kind of knowledge which acts a mechanism for situation presentation. The corresponding cue merging algorithm is as follows:

Algorithm 1 Cue Merging Algorithm

Input: A set of cues $\check{E}S = \{\check{E}_1, \check{E}_2, \ldots, \check{E}_n\}$

Output: A navigation knowledge: N

Procedure:

Step 1.Let $\check{E}_0 \leftarrow \varnothing$ and $i \leftarrow 1$.

Step 2.Get a cue $\check{E}_i$ from $\check{E}S$

Step 3.Get a causal relationship $r^c(c, c')$ in $\check{E}_i$

Step 4.If $r^c(c, c') \in \check{E}_0$, go to Step 5. Otherwise,

If $c \in \check{E}_0$, and $c' \notin \check{E}_0$, add a node c' then add a causal relationship $r^c(c, c')$ into $\check{E}_0$.

If $c \notin \check{E}_0$, and $c' \in \check{E}_0$, add a node c then add a causal relationship $r^c(c, c')$ into $\check{E}_0$.

If $c \notin \check{E}_0$, and $c' \notin \check{E}_0$, add two nodes c and c', then add a causal relationship $r^c(c, c')$ into $\check{E}_0$.

If $c \in \check{E}_0$, $c' \in \check{E}_0$, $r^c(c, c') \notin \check{E}_0$, add a causal relationship $r^c(c, c')$ into $\check{E}_0$.

If all causal relationships in $\check{E}_i$ have been processed, go to Step 5, otherwise go to Step 3.

Step 5.If $i < n$, $i \leftarrow i+1$, go to Step 2. Otherwise, go to Step 6.

Step 6.$N \leftarrow \check{E}_0$

Step 7.End.

As cues are represented as directed graphs (cue maps), the generation of navigation knowledge is a merge process of graphs. The algorithm first creates an empty cue $\check{E}_0$ and then tries to insert all cues in the cue set into $\check{E}_0$ one by one. For each cue to be inserted, it gets a causal relationship (edge) of this cue. If either of the two concepts (the two nodes of the edge) is not in $\check{E}_0$, the node and the corresponding causal relationship will be created in $\check{E}_0$. Note that, causal relationships are represented as directed edges. $r^c(c, c')$ and $r^c(c', c)$ are different causal relationships. Hence, if the two concepts already exist in $\check{E}_0$, but the causal relationship between them is $r^c(c', c)$, a new causal relationship $r^c(c, c')$ will be created.

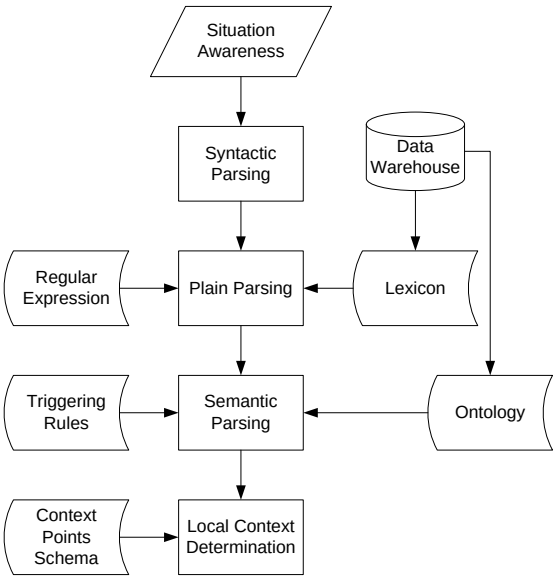


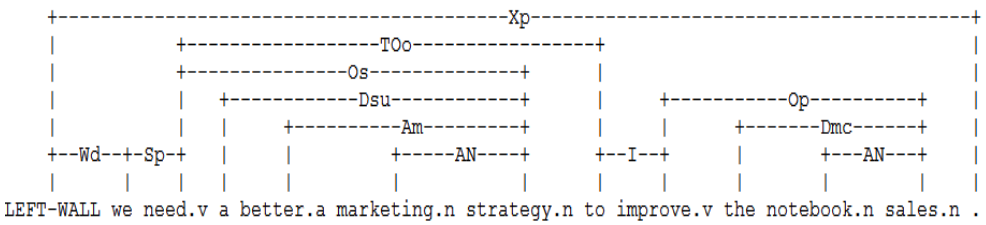
Figure 8 Situation Awareness Parsing

6. Situation Awareness Parsing

This section presents the process of SA parsing which includes four steps: syntactic parsing, plain parsing, semantic parsing and local context determination, as shown in Figure 8.

Step 1. Syntactic Parsing

In this research, we employ the Link Grammar Parser [32] to analyze managers' SA descriptions. The Link Grammar Parser is an English syntactic parser. Given a sentence, the parser generates a syntactic structure called *linkage* consisting of a number of *links*. An example of linkage is shown in Figure 9. Each link represents a syntactic relationship (denoted dashed lines in Figure 9) between two words. Links can be of different types. For instance, the link type between *We* and *need* is *Sp*. A *Sp* link connects a subject-noun (*We*) to a finite verb (*need*). Note that the Link Grammar Parser will insert an artificial word at the beginning and the end of every sentence before it is parsed, known as the *wall*.



Note: Xp simply connects the period to the left wall at the end of a sentence. Wd is used in ordinary declarative sentences, to connect the main clause back to the wall (or to a previous coordinating conjunction). Sp connects plural nouns to plural verb forms. TOo connects a verb to the word "to". Os and Op: O connects transitive verbs to direct or indirect objects. Os- and Op- connectors mark nouns as being singular or plural. Dsu and Dmc: D connects determiners to nouns. The first subscript place distinguishes between singular ("s") and everything else ("m"); the second place distinguishes between plural ("c") and mass

("u") (for "countable" and "uncountable"). AM is used in the comparative constructions "as much" and "as many". AN connects noun-modifiers to nouns. I connects certain verbs with infinitives.

Figure 9 Linkage Produced by Link Grammar Parser

Based on the linkage of a sentence, the Link Grammar Parser also chunks sentences into phrases and produces constituent trees. The corresponding constituent tree for the linkage is shown in Figure 10.

```
(S (NP We)
   (VP need
       (NP a better marketing strategy)
       (S (VP to
              (VP improve
                  (NP the notebook sales)))))
   .)
```

Figure 10 Constituent Tree Produced by Link Grammar Parser

Using the Link Grammar Parser, SA sentences can be chunked down into words or phrases. Meanwhile, different syntactic relationships between these words or phrases can be identified. Based on the result of the Link Grammar Parser and an ontology, relevant instances and classes of the ontology can be extracted or inferred. The words, phrases, instances and classes are semantically isolated. We refer to them as ***SA tokens***. If we use an SA token set to collect all SA tokens generated in the course of SA parsing, this SA token set initially only contains the words and phrases output by the parser. Then instances and classes are gradually added into the SA token set. By annotating SA tokens based on the domain ontology, SA triples and local contexts can be produced.

Step 2. Plain Parsing

Plain parsing recognizes meta instances from SA descriptions. There are two kinds of meta instances to be recognized: *numeric meta instances* and *literal meta instances*.Numeric meta instances are recognized using pattern matching techniques which are based on regular expressions. The following is an example of regular expressions:

"(*19*|*20*)[*0-9*][*0-9*]-(([*1-9*])|(*0*[*1-9*])|(*1*[*0-2*]))-(([*1-9*])|(*0*[*1-9*])|([*1-2*][*0-9*])|(*3*[*0-1*]))"

This regular expression is used to extract *date* information of format YYYY-MM-DD, where YYYY, MM, DD respectively represent a year of four digits, a month of two digits and a day of two digits, e.g., *2008-08-25*.

The recognition of literal meta instances from SA is based on lexicons. Generally speaking, literal instances are enumerable information. A domain-specific lexicon can be pre-created to cover all possible literal meta instances. An item (entry) in the lexicon contains four kinds of information: the value of a meta instance, the corresponding property, the corresponding class and information type[2]. The following is an example of lexicon item.

Lexicon item: *Mountain-100 Silver/EnglishProductName/Mountain Bike/0*

[2] We use the term information type to represent the different types of semantic information that SA instances carry. Examples of information types defined in this research are *area, speed, money* and *temperature*.

where,

Instance value: *Mountain-100 Silver*
Property: *EnglishProductName*
Class: *Mountain Bike*
Information Type: *0*

The searching for literal meta instances in the lexicon is a straightforward boolean matching process. If a match is found between an SA token and the *instance value* of an item in the lexicon, this lexicon item is exported as an SA triple. The corresponding SA triple is created by setting its context, view and wording as the class, property and value of the lexicon item respectively.

Step 3. Semantic Parsing

Based on matching techniques, plain parsing produces primitive semantic results, i.e., meta instances. While semantic parsing can infer new concepts based on the primitive results including the ontology, the SA tokens, the lexicon, the SA triples produced in plain parsing, and the linkage generated by the Link Grammar Parser. Semantic parsing infers ontology classes (both general and abstract classes) from SA and uses these classes to reduce the uncertainties of SA triples based on ***class triggers***. A class trigger is associated with a specific class. The presence of a class trigger in an SA sentence and/or SA tokens might result in the presence of this class. However, a class trigger does not guarantee that a class will be successfully inferred, unless pre-defined rules are met at the same time. Examples of class trigger for PRODUCT RELEASE are as follows.

{ *product, release* }
{ *product release* }
{ *product, announcement* }

As can be seen from the above examples, a class trigger is a term set consisting of a number of terms, each of which can be a class, a word or a phrase. There are four types of class triggers.

- **Single-Element Class Triggers**

The triggering rule for single-element class triggers is as follows.

Triggering Rule 1: *The element of a class trigger can be matched with one of the SA tokens***.**

This triggering rule is very simple: as long as a class trigger can be found among the SA tokens, the corresponding class can be inferred. Let us look at some examples.
Suppose an ontology is shown in Figure 3 and an SA sentence is as follows:

We released Mountain-100 Silver in Australia in 2005.

Algorithm *LiteralPlainParser* can generate an SA triple from this SA sentence:

(MOUNTAIN BIKE, English Product Name, *Mountain-100 Silver*).

The SA token *Mountain-100 Silver*, is a meta instance of the taxonomy property of MOUNTAIN BIKE. English Product Name refers to the corresponding property of the class. According to the taxonomy-property-based method for class trigger construction, {*Mountain-100 Silver*} is a class trigger of MOUNTAIN BIKE and it is also a single-element class trigger. Class MOUNTAIN BIKE can be triggered (inferred) by the SA token *Mountain-100 Silver*, i.e.,

Mountain-100 Silver → *MOUNTAIN BIKE*.

In Figure 3, class MOUNTAIN BIKE is a child class of BIKE; BIKE is a child class of PRODUCT.

According to the child class based method, {*mountain bike*} is a class trigger of BIKE; {*bike*} is a class trigger of PRODUCT. Thus, classes BIKE and PRODUCT can also be inferred by

MOUNTAIN BIKE → *BIKE*, and *BIKE* → *PRODUCT*.

- **Multi-Element Class Triggers**

Multi-element class triggers have two basic varieties and they correspond to different triggering rules.

(I) Noun-Noun Class Triggers

This type of class trigger consists of two elements, both of which are nouns, for example, {marketing, strategy} and {internet, sales}. The corresponding triggering rule is as follows:

Triggering Rule 2: *The two nouns of a class trigger appear in a noun phrase which has the same head noun as the class trigger.*

We use an example to illustrate this triggering rule.

Suppose an SA sentence is as follows:

The release of Mountain-100 Silver increased our market share by 45%.

The triggering process has four steps as follows:

Step 1. Triggering Rule 1: *Mountain-100 Silver* → *MOUNTAIN BIKE*

Step 2. Triggering Rule 1: *MOUNTAIN BIKE* → *BIKE*

Step 3. Triggering Rule 1: *BIKE* → *PRODUCT*

Step 4. Triggering Rule 2: *release, PRODUCT* → *PRODUCT RELEASE*

The first three steps are based on Triggering Rule 1. In Step 4, PRODUCT is not actually present in the SA sentence. However, looking back at the process of how PRODUCT is triggered, algorithm *SemanticParser* will replace *Mountain-100 Silver* in the SA sentence with PRODUCT. Triggering Rule 2 is met after this replacement: (1) *release of product* will be output as a noun phrase by the Link Grammar Parser; (2) *release* is the head noun of this noun phrase and the class trigger {*product, **release***}. Therefore, class PRODUCT RELEASE will be successfully triggered by class trigger {*product, **release***}.

(II) Verb-Noun Class Triggers

This type of class triggers consist of two elements: a verb and a noun, for example, {*release, product*} and {*improve, sales*}. There are two corresponding triggering rules as follows.

Triggering Rule 3: *The noun is the direct object of the verb.*

Triggering Rule 4: *The noun is the subject and the verb is a passive participle.* .

We use an example to illustrate Triggering Rule 3.

Suppose an SA sentence is as follows:

We released Mountain-100 Silver in Australia in 2005.

The triggering process is as follows:

Step 1. Triggering Rule 1: *Mountain-100 Silver* → *MOUNTAIN BIKE*

Step 2. Triggering Rule 1: *MOUNTAIN BIKE* → *BIKE*

Step 3. Triggering Rule 1: *BIKE* → *PRODUCT*

Step 4. Triggering Rule 3: *released, PRODUCT → PRODUCT RELEASE*

In Step 4, *Mountain-100 Silver* is replaced by PRODUCT, such that PRODUCT is the direct object of *released*. Triggering Rule 3 is met and class PRODUCT RELEASE is triggered.

The triggering process for Triggering Rule 4 is similar to Triggering Rule 3. Based on this rule, class PRODUCT RELEASE can be triggered from the following SA sentence.

Mountain-100 Silver was released in Australia in 2005.

The presence of a class trigger will not trigger the class if the corresponding triggering rule is not met. Look at the following two SA sentences,

*Another marketing campaign was **released** after the sales of **Mountain-100 Silver** went down.*

*Occasionally, **Mountain-100 Silver releases** oil when climbing steep ramps.*

As none of the triggering rules can be met, class PRODUCT RELEASE cannot be triggered, although the class triggers {released, product} and {product, releases} are present.

The single-element and two-element class triggers correspond to single-word and two-word classes respectively. In an ontology, one-word classes are generally the most common type. Of multi-word classes, two-word classes are the most common type. There are very few classes with the number of words over three. For example, the maximum length (the number of words in class labels) in the Enterprise Ontology [33] is three. There are 62 single-word classes (63%), 34 two-word classes (34%) and 3 three-word classes (3%). Of multi-word classes, 92% are two-word classes and only 8% are three-word classes. In the ontology developed in the experiment, there are 69 single-word classes (63%), 36 two-word classes (33%) and 5 three-word classes (less than 5%). Therefore, the four triggering rules which have been discussed can deal with about 96% cases of class triggering (in both ontologies). For class triggers whose lengths are three or longer, the corresponding triggering rules become more complex. For example, a class trigger might consist of a verb and a number of nouns. One corresponding triggering rule is as follows:

All nouns appear in a noun phrase and one of them is the head of this phrase. This noun phrase is the direct object of the verb.

Step 4. Local Context Determination

The local context of an SA sentence represents the common background where the majority of SA triples are communicated. All classes which are inferred during semantic parsing are alternatives for the local context. We use a point system to select the local context from these alternatives.

In this point system, the points of an alternative context reflect its competitiveness to be selected as the local context. We refer to theses points as its ***context points*** (***CP***). We defined three metrics to evaluate the context points of an alternative context from three aspects: ***context coverage point*** (***CCP***), ***context position point*** (***CPP***) and ***inverse context specificity point*** (***ICSP***).

$$CP = CPP + CCP + ICSP.$$

The *CPP* points of an alternative context are related to its syntactic position in the SA sentence. In linguistics, a complete English sentence is divided into smaller chunks, i.e., POS (part of speech) based on their syntactic positions. From a semantic point of view, different POS signifies different semantic information within the context of the sentence. Points can thus be assigned to each POS to signify the degree to which that POS complies with the context of the sentence. We use CPP to represent this sort of points (Table 1). Please note that the CPP points in Table 1 are given on a scale of 1–5 based on our experience and can be adjusted according to application domains.

Table 1 Point Schemas for Context Position and Context Coverage

Position	Part of Speech	CPP	CCP
1.	pre-subject adjective modifier	2	1
2.	pre-subject noun modifier	3	1
3.	head noun (subject)	5	1
4.	post-subject modifier (prepositional phrase/relative clause)	3	1
5.	head verb (predicator)	4	1
6.	verb modifier	1	1
7.	pre-object adjective modifier	1	1
8.	pre-object noun modifier	2	1
9.	direct/indirect object	3	1
10.	post-noun modifier (prepositional phrase/relative clause)	2	1

The *CCP* points of an alternative context are related to its coverage across the SA sentence. As part of a SA sentence, different alternative contexts cover different number of POS. The local context should have a wider coverage within the sentence. Different POS have an equal CCP point which is shown in Table 1.

The *ICSP* points of an alternative context are related to its relative position in the class tree. Classes in different positions in the class tree correspond to different class specificities. We assume an alternative context with lower specificity (more abstract) in the ontology is more competitive to be the local context of this sentence. We use term ***class specificity*** to describe the degree to which a class conveys detailed information. Classes from the root of the class tree down to leaf node classes have different specificities. The closer to a leaf node, the higher specificity a class has. This kind of class specificities are determined by distances to the nearest leaf node (***DTL***) in the class tree. Let *c* be a class. We use the following formula to compute *DTL* of *c*:

$$DTL = 1 + d,$$

where d is the distance from class *c* to the nearest leaf node.

The specificity of a class is also related to the semantic parsing level. A general class is inferred by the meta instances of its taxonomy property. We refer to the semantic parsing to infer general classes as level 1 parsing. On the basis of general classes, some abstract classes can be inferred, which we call level 2 parsing. Similarly, classes inferred at level 2 parsing lead to level 3 classes, and so on. There are situations where an abstract class is inferred from multiple lower level classes, i.e. the trigger members are distributed in different positions in the class tree. In this case, we assess the parsing level by averaging all lower parsing levels. We use ***class abstract level*** (***CAL***) to refer to this kind of class specificities.

The inverse concept specificity points are computed by the following formula

$$ICSP = DTL + CAL.$$

Let us look at an example to show the complete process of local context calculation.

Assume we have a SA Sentence:

We released Mountain-100 Silver in Australia in 2005

Alternative Contexts (generated in syntactic parsing process):

PRODUCT, PRODUCT RELEASE

The inferring process for class PRODUCT is

Mountain-100 Silver → MOUNTAIN BIKE → BIKE → PRODUCT

Thus, PRODUCT can be tracked back to *Mountain-100 Silver* falling in Position 9 (direct/indirect object) (Table 1). The PRODUCT earns 3 *CPP* points. Similarly, PRODUCT RELEASE falls in Position 5 (head verb (predicator)) and Position 9 (direct/indirect object). Thus PRODUCT RELEASE has 7 *CPP* (4 + 3) points.

According to the *CCP* schema defined in Table 1, PRODUCT covers only one POS (Position 8). It earns 1 *CCP* point. PRODUCT RELEASE earns 2 *CCP* points.

Based on the ontology (Figure 17) used in application case in the Appendix, the *DTL* values for PRODUCT and PRODUCT RELEASE can be calculated as follows.

DT L (PRODUCT) = 3

DTL (PRODUCT RELEASE) = 1.

CAL (PRODUCT) = 3

CAL (PRODUCT RELEASE) = 4

Thus, the ICSP values are as follows.

ICSP (PRODUCT) = 3+3 = 6

ICSP (PRODUCT RELEASE) = 1+ 4 = 5.

The *CP* points of an alternative context are the sum of its *CPP* points, *CCP* points and *ICSP* points,

CP (PRODUCT) = 3 + 1 + 6 = 10

CP (PRODUCT RELEASE) = 7 + 2 + 5 = 14

The local context of SA sentence

We released Mountain-100 Silver in Australia in 2005

is PRODUCT RELEASE.

7. Query Construction

Query construction and situation presentation is based on the mapping relationships between SA, experience base, ontology, and data warehouse and situation, which are illustrated in Figure 11. There are five layers in this figure separated by horizontal lines.

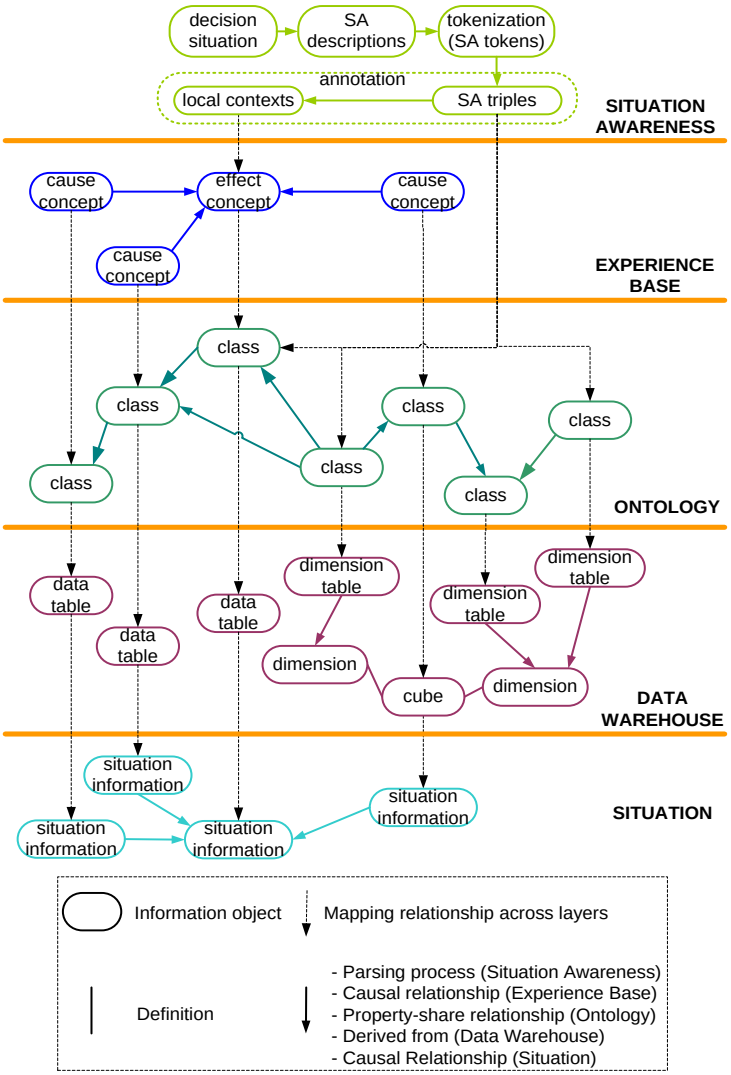


Figure 11 Query Construction and Situation Presentation

(1) Situation Awareness

In the SA layer, the manager describes his/her SA about the current decision situation and inputs into the system in the form of natural language. Each SA description is comprised of a number of SA sentences. SA sentences are parsed into SA tokens. Finally, SA triples are generated by annotating the SA tokens, and local contexts of SA sentences are also determined. The SA triples and local contexts are mapped into the ontology and the experience base.

(2) Experience Base

In this layer, the manager's experiences are represented as experience maps. An experience map consists of two sorts of concepts (cause concepts and effect concepts) and the causal relationships between them.

The local contexts of SA sentences are used to generate a knowledge need, which are then used to extract cues from the experience base. The extracted cues represent a fragment of the entire experiences, which imply potential solutions, ideas or relevant situation information. In the process of knowledge retrieval, a local context in the SA layer is directly mapped to an effect concept in the experience base layer.

(3) Ontology

Classes are defined in the ontology layer. Each class is described by a set of properties. Classes are associated with each other via two sorts of relationships: subsumption relationships and property-share relationships.

An SA triple in the SA layer is mapped to a combination of a class, a property of this class and a meta instance of this property, in the ontology layer. A property-share relationship enables semantic information expressed by SA triples to be transferred from a class to another class. In this way, those classes, which do not have direct mapping relationships with SA triples, can also be indirectly mapped.

(4) Data Warehouse

In the data warehouse layer, situation information is hiding in either data tables (relational data) and/or cubes (multidimensional data). The objective of query construction is to find the situation information for situation presentation. A cube is defined by a number of dimensions and measures. Dimensions are derived from dimension tables.

A class in the ontology layer can be mapped to a data table, a cube or a dimension table in the data warehouse layer. The mapping relationship is defined during the development of the ontology.

Queries are constructed and executed against the data warehouse. During the execution of the queries, situation information is retrieved from data tables or cubes. The retrieved situation information is transferred to the situation layer for situation presentation.

(5) Situation

In this layer, the original decision situation is presented to the manager through displaying the retrieved situation information. Situation is presented through different data visualization techniques, such as graphs, charts and tables. When the manager perceives and understands the presented situation information, his/her SA is developed and enriched.

The process of query construction is illustrated in Figure 12. It includes two basic tasks: determining query data sources and constructing query clauses.

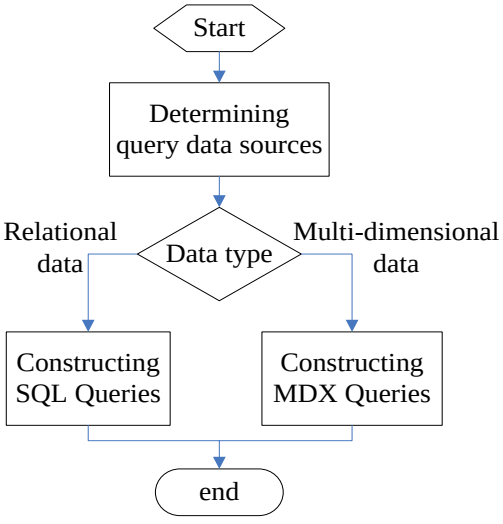


Figure 12 Flowchart of query construction

The procedure of determining the data source for a query is as follows.

Step 1. The navigation knowledge is visualized as a map and presented to the manager.

Step 2. The manager/user (decision maker) browses the navigation knowledge map and expresses interests in a specific concept.

Step 3. A class which is matched with the concept of the manager's interest is found in the class tree of the ontology.

Step 4. If the class found in Step 3 has an associated qualified DW object (data table or cube), output this DW object as the query data source. Otherwise, find the most similar qualified class through assessing class similarities and output the DW object of this new class as the query data source.

To deal with relational data, the algorithm ***SqlBuilder*** is developed to construct SQL queries. *SqlBuilder* intakes a concept of the navigation knowledge and outputs the corresponding SQL query for this concept.

Algorithm 2 SqlBuilder

Input:

(1) A navigation knowledge concept: ***c***

(2) A set of SA triples: ***SATripleSet***

(3) An ontology

Output:

A SQL query for concept ***c***

Procedure:

See Figure 13.

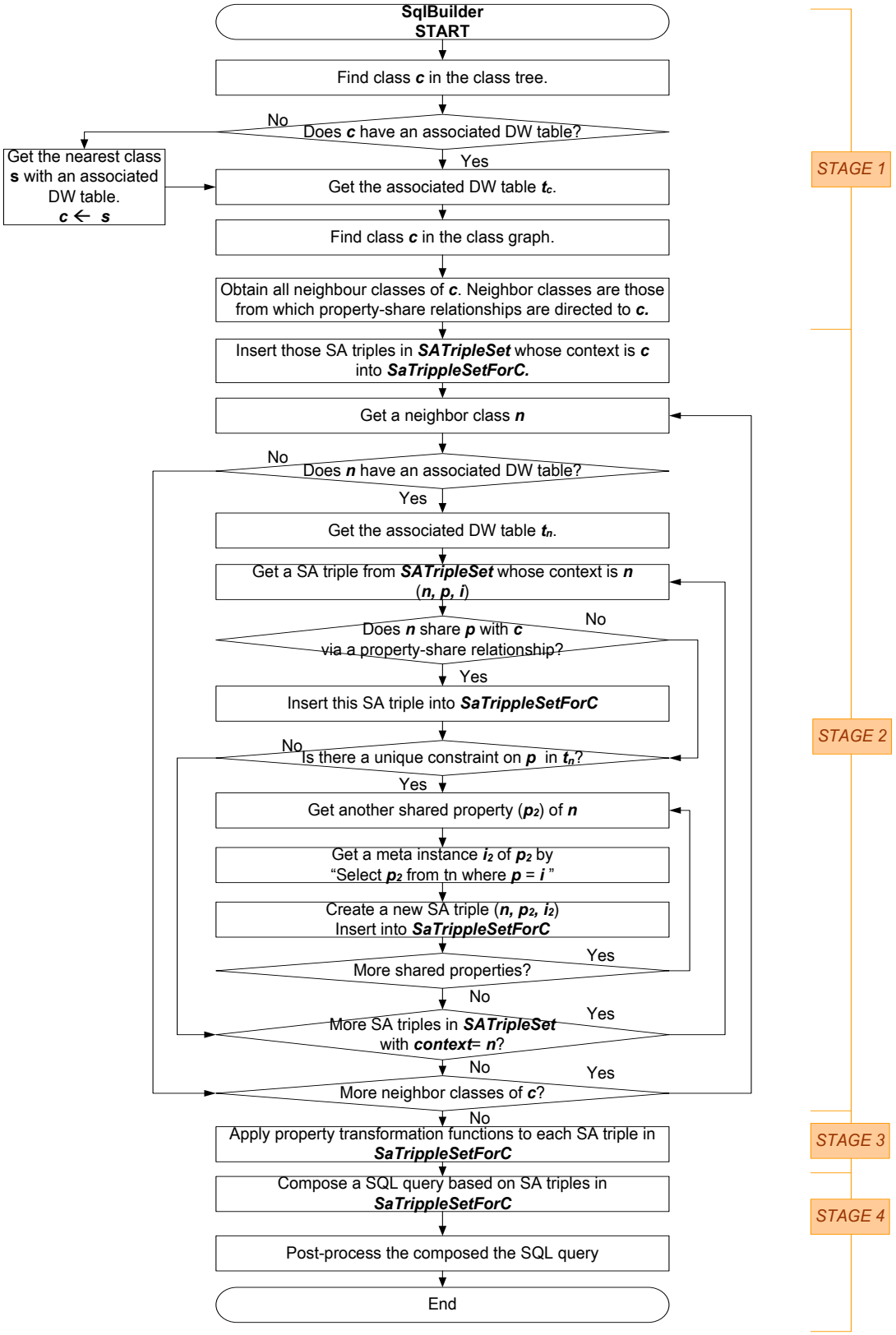


Figure 13 Algorithm: SQLBuilder

For example, based on the following SA triple group

{(*Mountain Bike, EnglishProductName, Mountain-100 Silver*),

(*Mountain Bike, EnglishProductName, Mountain-100 Black*)},

two predicates are created as follows.

EnglishProductName = Mountain-100 Silver

EnglishProductName = Mountain-100 Black

The two predicates are connected using the logical operator *OR*.

EnglishProductName = Mountain-100 Silver OR EnglishProductName = Mountain-100 Black

Predicates created based on different SA triple groups are connected using the logical operator *AND*. For example, based on SA triple

(*Mountain Bike, color, red*)},

a predicate is created as follows.

Color = red

Connected with the previous two predicates using AND,

(*EnglishProductName = Mountain-100 Silver OR EnglishProductName = Mountain-100 Black*) *AND*

Color = red

Thus, a WHERE clause can be composed as follows.

WHERE (*EnglishProductName = Mountain-100 Silver OR EnglishProductName = Mountain-100 Black*) *AND Color = red*

Finally, a provisional SQL query is constructed as follows.

SELECT * FROM $\boldsymbol{t_c}$ WHERE (*EnglishProductName = Mountain-100 Silver OR EnglishProductName = Mountain-100 Black*) *AND Color = red*

After post-process, *SqlBuilder* will output the following query.

SELECT * FROM $\boldsymbol{t_c}$ WHERE (*EnglishProductName =' Mountain-100 Silver' OR EnglishProductName = 'Mountain-100 Black'*) *AND Color = 'red'*

Note that *Mountain-100 Silver* and *Mountain-100 Black* are enclosed with quotation marks according to their data type (*nvarchar*) in the definition of $\boldsymbol{t_c}$.

To deal with multi-dimensional data, the algorithm ***MdxBuilder*** is designed to construct MDX queries. *MdxBuilder* intakes a concept of a navigation knowledge and outputs the corresponding MDX query.

Algorithm 3 MdxBuilder

Input:

(1) A navigation knowledge concept: ***c***

(2) A set of SA triples: ***SATripleSet***

(3) An ontology

Output:

A MDX query for concept ***c***

Procedure:

See Figure 14 and Figure 15.

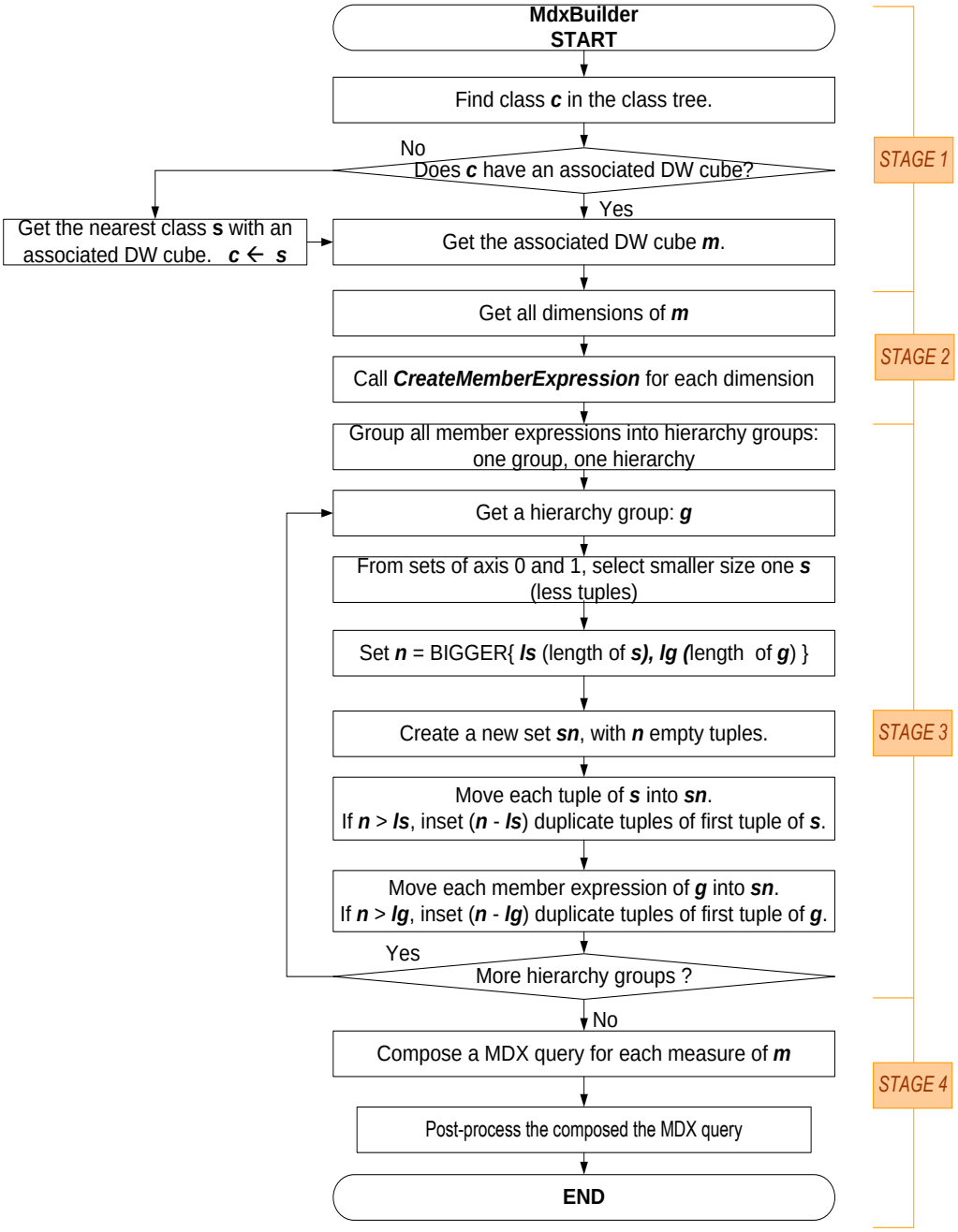


Figure 14 Algorithm MDXBuilder

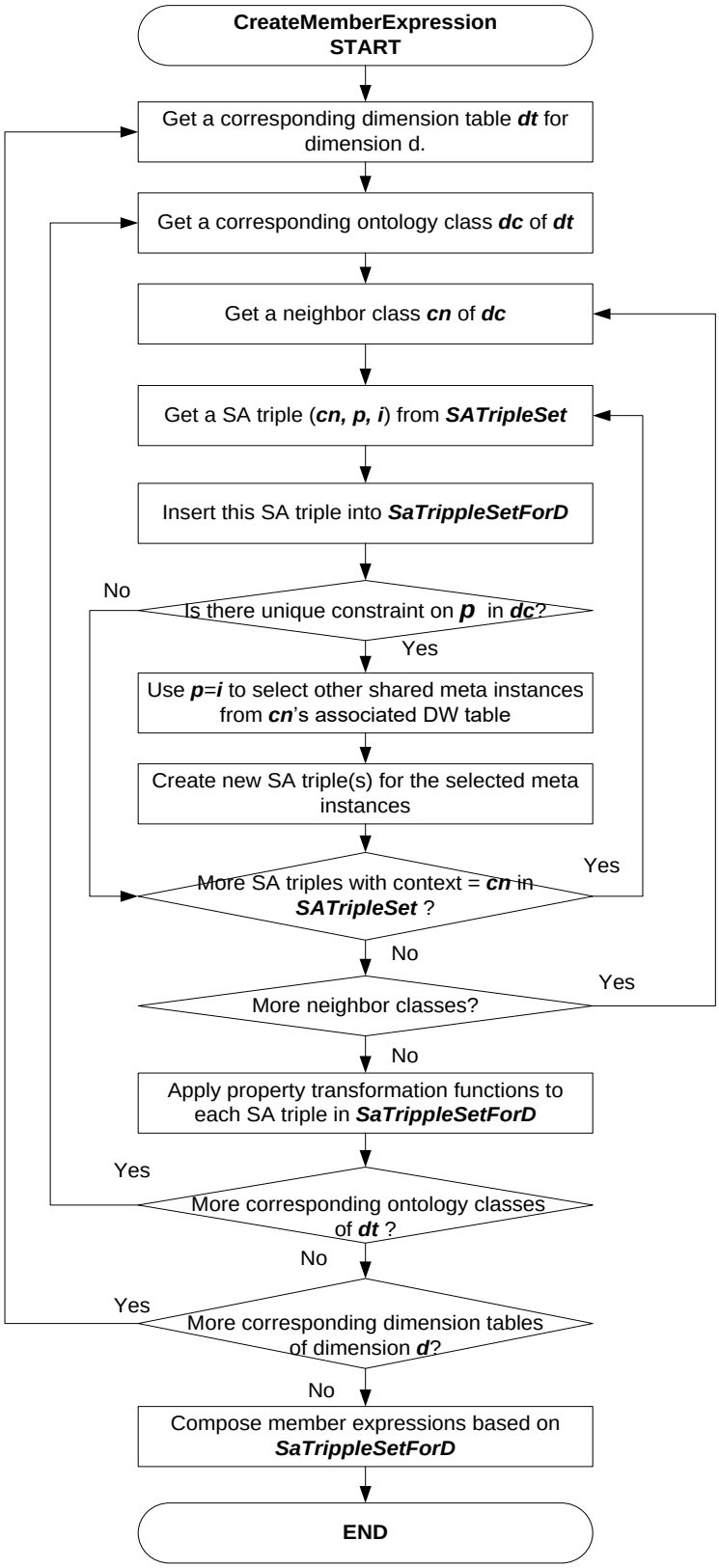


Figure 15 Algorithm CreateMemberExpression

An example is given as follows:

SA Triples:

(*Mountain Bike, EnglishProductName, Mountain-100 Silver*)
(*Road Bike, EnglishProductName, Road--30 Black*)
(*Mountain Bike, Color, Blue*)
(*Customer, Location, France*)
(*Customer, Location, Canada*)

Member Expressions which are composed based on the above SA triples:

[*Product*].[*Model Name*].[*Mountain-100 Silver*]
[*Product*].[*Model Name*].[*Road--30 Black*]
[*Product*].[*Product Color*].[*Blue*]
[*Customer*].[*Country*].[*France*]
[*Customer*].[*Country*].[*Canada*]

Based on the member expressions generated above, two MDX sets can be constructed as follows.

Axis 0 set:

{([*Product*].[*Model Name*].[*Mountain-100 Silver*], [*Product*].[*Product Color*].[*Blue*]),
([*Product*].[*Model Name*].[*Road--30 Black*], [*Product*].[*Product Color*].[*Blue*]) }

Axis 1 set:

{([*Customer*].[*Country*].[*France*]), ([*Customer*].[*Country*].[*Canada*])}

Finally, a complete MDX query is constructed by assembling tuples, the query cube and measure. MdxBuilder constructs a MDX query for every measure of a cube. Let ***r*** be a measure of cube ***m***. The following is an example of a finished MDX query.

SELECT
{([*Product*].[*Model Name*].[*Mountain-100 Silver*], [*Product*].[*Product Color*].[*Blue*]),
([*Product*].[*Model Name*].[*Road--30 Black*], [*Product*].[*Product Color*].[*Blue*]) } ON 0,
{([*Customer*].[*Country*].[*France*]), ([*Customer*].[*Country*].[*Canada*])} ON 1
FROM [*m*]
WHERE [*r*]

Note that in the above MDX query, measure ***r*** is used as a WHERE clause, instead of the member of the special dimension *Measure*.

8. Situation Presentation

We use navigation knowledge to guide the process of situation presentation. This method is depicted in Figure 16. The relationships between situation information and environment data are established by data warehouse (DW) queries. When DW queries are executed against the data warehouse, corresponding situation information can be retrieved. A piece of situation information might correspond to different data in the data warehouse. In Figure 16, solid directed lines are used to denote these direct relationships.

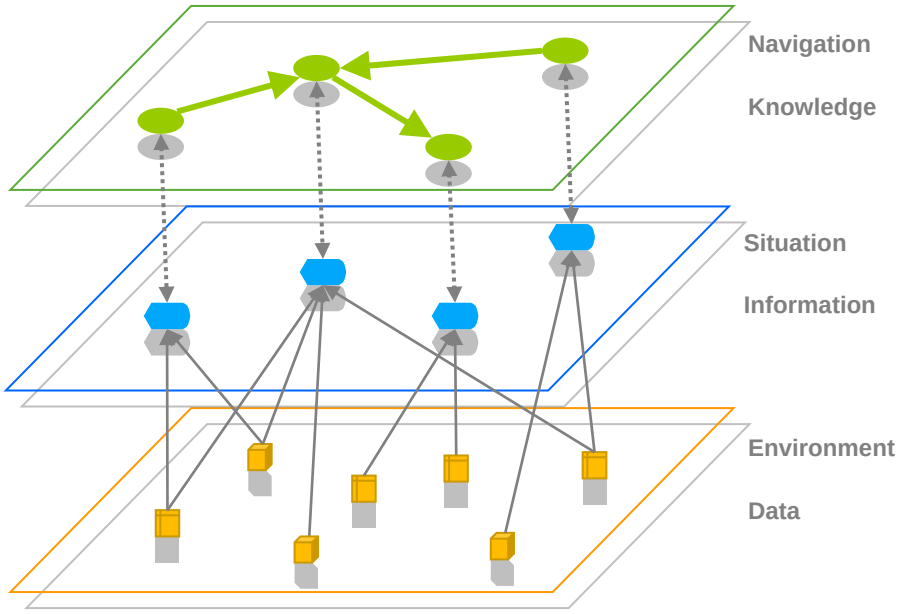


Figure 16 Navigation-Knowledge-Guided Situation Presentation

The dashed directed lines denote indirect relationships between situation information and navigation knowledge. Navigation knowledge is visualized as maps on a graphical user interface (GUI). The representation of navigation knowledge also consists of concepts and causal relationships. Each concept implies a kind of information which reflects an aspect of the decision situation. If the manager is interested in a specific concept in the navigation knowledge, he/she can request the system to show related situation information. For example, the request can be made by mouse-clicking onto a concept on the navigation knowledge map. When the system receives an information request, the corresponding queries will be triggered and executed against the data warehouse. Relevant data is then retrieved from the data warehouse (environment data). The retrieved data is processed using data analysis techniques and then presented to the manager on the GUI. The general pattern of human-machine interaction in the navigation-knowledge guided situation presentation is as follows.

> *A navigation knowledge map is presented →(The manager) Browse the experience → Find a concept interesting → Click this concept for associated information → Detailed information is presented → Perceive the detailed information → Return back to the navigation knowledge map → Browse the experience → Find another concept interesting →* **…**

The navigation knowledge is the combination of all cues relevant to the current decision situation. Cues prompt possible ideas, clues or solutions to a current decision situation. A cue is retrieved according to a concept (an element of the knowledge needs related to the current decision situation) from the experience base. A decision situation might have many related concepts. A cue is generated only based on one concept. Hence, a cue can only reflect part of the decision situation. The navigation knowledge is the integration of all related cues of a decision situation. Therefore, the visualization of navigation knowledge presents a complete *big picture* of the current decision situation to the manager. The navigation-knowledge-guided situation presentation also allows managers to add extra concepts and causal relationships to the generated navigation knowledge. For example, suppose the navigation knowledge shows that PRODUCT and ADVERTISEMENT are the key factors to SALES. If a manager thinks GOVERNMENT POLICY is also playing a role, he/she can supply this new concept

and the relationship with SALES to the system. The navigation knowledge map and the corresponding DW queries will be updated accordingly.

9. Experiments

This section presents experiment results of the FACETS.

9.1 Experiment Design

We use two basic metrics to measure the performance of algorithms: precision and recall. Precision and recall are initially proposed for evaluating the quality of searching results of information retrieval systems in terms of information relevance [34]. Precision is defined as the fraction of the relevant documents within the collection of all retrieved documents. Recall is the fraction of the documents within the collection of all relevant documents from the data source.
Let r be the number of relevant documents returned by the system, i be the number of irrelevant documents returned by the system, and m be the number of relevant documents in the data source that are not retrieved by the system. The formulas for computing precision and recall are as follows [34].

$$Precision = \frac{r}{r+i}$$

$$Recall = \frac{r}{r+m}$$

We also evaluated FACETS in terms of its usefulness and usability. Usefulness and usability are two dimensions of information value defined by Rouse [25]. The usefulness of a decision support system refers to the extent to which information produced by the system can help users make decisions. The usability is the extent to which information can be easily accessed, understood and applied for decision making. We used a scenario-based experiment to evaluate the usefulness and usability of FACETS.
A decision scenario was created for this experiment. The decision scenario set up a decision situation where each subject in the experiment participated in a decision-making process based on FACETS. Subjects described their SA using English and input into FACETS. FACETS called corresponding algorithms to analyze SA, generate DW queries, retrieve relevant situation information and present situation information to the subjects. Perceiving situation information, the subjects updated their SA and re-input into FACETS. In this way, each subject went through up to five decision cycles with the purpose of working out an appropriate decision for the business problem presented in the decision scenario. However, the goal of the experiment was not to evaluate the outcome of decision generation during a cognition-driven decision process, but to collect subjective data about their decision-making processes. The subjective data was used to evaluate the usefulness and usability of FACETS. During decision cycles, subjects were requested to fill out an evaluation form about query construction.
In the experiment, each subject ran through up to five decision cycles one by one. During each decision cycle, FACETS generated SQL/MDX reports to present situation information. A SQL report contained data retrieved from a relational table. Subjects browsed the report and judged its usefulness in terms of decision making. An MDX report contained data retrieved from a cube which included a number of dimensions. Subjects also judged the usefulness of dimensions in term of decision making. At the end of each decision cycle, subjects were required to answer three questions:

(1) *How many relational tables/dimensions did FACETS output which you think are useful for your decision situation? The answer is the number of accepted tables/dimensions.*
(2) *How many relational tables/dimensions did FACETS output which you think are useless for your decision situation? The answer is the number of rejected tables/dimensions.*
(3) *How many relational tables/dimensions did FACETS miss (fail to find) which you think are useful for your decision situation? The answer is the number of missed tables/dimensions.*
At the end of the decision process, subjects filled out a survey form to rate FACETS as a whole. The survey form included twenty-four questions which were designed to elicit subjects' opinions about FACETS. These questions could be divided into two categories: usefulness evaluation and usability evaluation. Subjects were asked to give their subjective ratings for each question on a scale of five (*1: strongly disagree, 2: disagree 3: neutral, 4: agree, 5: strongly agree*).

9.2 Query Construction Evaluation

Based on the evaluation forms of query construction that subjects filled out in the experiment, the performance of query construction was evaluated and is shown in Table 2. The precision and recall for each decision cycle were calculated.

Table 2 Query Construction Performance

Decision Cycles	1	2	3	4	5
Average # of Accepted Tables/Dimensions	0.9	4.0	8.5	9.2	9.5
Average # of Rejected Tables/Dimensions	1.7	3.1	3.4	3.3	3.6
Average # of Missed Tables/Dimensions	6.7	5.9	4.4	2.1	1.1
Precision	0.35	0.56	0.71	0.74	0.73
Recall	0.12	0.40	0.66	0.81	0.90

Table 2 shows that the average number of tables/dimensions accepted by subjects jumps up rapidly from 0.9 to 8.5 over the first three decision cycles. SA is accumulated over decision cycles. The more iterations of decision cycle, the more SA that subjects input into FACETS and consequently FACETS found more relevant situation information. In this sense, the iteration of decision cycles is valuable for decision making. However, the speed of increase in the average number of accepted tables/dimensions slows after a number of decision cycles. For example, there is only marginal increase of the average number of accepted tables/dimensions during decision cycles 4 and 5. Note that SA was accumulated in FACETS over decision cycles. The information retrieved by FACETS based on SA in a decision cycle might be repeatedly retrieved and presented in a later decision cycle as long as the subject did not change or delete the corresponding SA.
Accompanying the increase of accepted tables/dimensions over decision cycles, the average number of rejected tables/dimensions also goes up, although a slight fall can be seen in decision cycle 4. This means that noise information always comes with valuable situation information in FACETS. In decision cycle 1, there are more tables/dimensions rejected than accepted. After that, accepted tables/dimensions always outnumber the rejected ones. The gap in the quantity of the two sorts of data widens over the decision cycles. This shows that FACETS is able to quickly locate much relevant

information and filter noise data for decision making, which can also be seen from the trend of precision change. Precision goes up over the decision cycles. From decision cycle 1 to decision cycle 5, the precision is increased by 0.38.

Over the decision cycles, there are also noticeable changes in the average number of missed tables/dimensions. Less tables/dimensions relevant to decision making were missed by FACETS when it took more SA input. This pattern is also reflected by recall which rises continuously from 0.12 in decision cycle 1 to 0.90 in decision cycle 5.

Based on the above analysis, both the quantity (accepted tables/dimensions) and quality (precision and recall) of information output by FACETS increase significantly over decision cycles. Different stages (decision cycles) of the decision process have various growing speeds: higher in the initial stage, e.g., the first three decision cycles in this experiment, and lower in the latter stage. Possible reasons for the quantity and quality of situation information to stop rising over the decision cycles are (1) newly input SA does not make a significant difference to existing SA; (2) the limitation of knowledge base or the DW is reached. In the case of the reason (1), the decision maker needs to revise the SA description to stimulate FACETS to search for new information. For reason (2), either the knowledge base or the DW needs to be extended by feeding fresh data. The growing speed of information quantity and quality can also act as a type of resource limit discussed in Part I, which triggers a final decision to be made. That is to say, if no more new situation information can be found, then probably it is time to make the final decision.

Table 3 FACETS Performance

STATEMENTS	Evaluation Category [3]	Mean Rating	Standard Deviation
SITUATION AWARENESS			
1. I can precisely describe my SA for a given decision situation, using natural language.	2	4.10	0.67
2. I can easily input my SA into the system.	2	4.92	0.32
NAVIGATION KNOWLEDGE MAPS			
3. Navigation knowledge maps reflect my past thinking (reasoning) processes in decision processes.	1	4.41	0.53
4. Concepts in the navigation knowledge maps are related to my current SA.	1	3.87	0.82
5. Navigation knowledge maps help me to understand the current decision situation.	1	3.35	0.65
6. Navigation knowledge maps have implications for seeking relevant situation information.	1	4.56	0.54
7. Navigation knowledge maps have implications for making decisions in the current decision situation.	1	4.01	0.63
8. I can intuitively make sense of the presentation of a navigation knowledge map (understand its concepts and relationships).	2	4.79	0.51
SITUATION INFORMATION PRESENTATION			

[3] Usefulness:1; Usability: 2

9. The generated reports help me to understand the basic characteristics of the current decision situation.	1	4.68	0.34
10. The generated reports help me to gain in-depth insights into the current decision situation.	1	4.69	0.79
11. The generated reports help me to develop richer SA.	1	3.77	0.81
12. The generated reports help me to seek further situation knowledge.	1	3.01	0.45
13. The generated reports help me to seek further situation information.	1	4.75	0.66
14. The generated report helps me to make final decisions.	1	2.01	0.80
15. The navigation from the presentation of navigation knowledge maps to corresponding reports can be easily followed.	2	4.72	0.30
16. The reports can be easily understood.	2	4.89	0.33
OVERALL			
17. The user interface of FACETS is user friendly.	2	5.00	0.29
18. FACETS helps me to reduce mental workload during the decision process.	1	4.61	1.46
19. FACETS helps me to reuse my past management experience.	1	4.50	0.54
20. FACETS helps me to digest information more easily.	1	3.97	0.42
21. FACETS helps me to obtain valuable information more efficiently.	1	4.03	1.03
22. FACETS helps me to obtain valuable information more effectively.	1	4.67	0.87
23. FACETS helps me to make decisions more rapidly.	1	2.56	0.80
24. FACETS helps me to make decisions more confidently.	1	3.00	0.64

9.3 FACETS Evaluation

Based on the survey forms that subjects filled out during the decision scenario, the performance of FACETS as a whole was evaluated (Table 3). In Table 3, the mean rating and standard deviation (SD) for each question (statement) in the survey form were calculated. Questions were grouped into four categories: *situation awareness*, *navigation knowledge map*, *situation information presentation* and *overall*. Each question evaluates FACETS from either an information usefulness perspective or from an information usability perspective.

In questions (Q) 1 and 2 of Table 3, subjects agreed, using natural language, that they could easily describe their SA and input into FACETS. The natural language interface is important for cognitive decision support. Instead of asking for specific key words in traditional IR systems, FACETS allows managers to describe their SA using natural language. The natural language description of SA can be abstract or very specific. Using natural language, managers can very freely describe and input whatever they think is of significance to their decision making. This might encourage elicitation of valuable information from managers' cognitive processes, which is important to facilitate cognitive decision support.

The overall rating of FACETS in navigation knowledge map evaluation is 4.17. This reflects that the navigation knowledge map is an effective way to present experience for knowledge reuse in decision

making. In particular, navigation knowledge could reflect subjects' past thinking processes (Q3 with average rating of 4.41), and the presentation of navigation knowledge maps is easily followed (Q8 with average rating of 4.79). FACETS presents situation information with the guidance of navigation knowledge. Complex decision situations often involve a large quantity of relevant information. Faced with a large amount of information, managers are vulnerable to missing their clues without navigation. In FACETS, concepts of interest to decision situations are connected with situation information in navigation knowledge maps. Managers can always re-orientate themselves in the light of the navigation knowledge.

FACETS received an overall rating of 4.07 in situation information presentation evaluation. Subjects agreed that the situation information retrieved by FACETS was of high usefulness in other aspects (Q9 – Q13) and of high usability in Q15 and Q16. However, subjects in the experiment did not think that the situation information retrieved by FACETS could really help them work out a final decision, for example, Q14 only scored an average of 2.01, which is the lowest rating in this survey. A possible explanation is that the decision scenario was created based entirely on fictitious business data which lacks the sufficient ability to imply real business patterns. Some data might conflict with the subjects' commonsense and existing knowledge about business.

In the overall evaluation of FACETS, Q17 received the highest rating (5.0): all subjects were comfortable with the interface of FACETS. Subjects thought their mental workload in decision making was reduced by using FACETS (Q18/4.61). FACETS helped subjects to reuse historical experience in the current decision situation (Q19/4.5). FACETS also improved the efficiency and effectiveness of obtaining valuable situation information (Q21/4.03 and Q22/4.67). Again, FACETS was rated lowly in questions related to actual decision results (Q23 and Q24).

In terms of usefulness, the overall rating of FACETS is 3.91. Thus, the information presented by FACETS is helpful for decision makers to develop relevant SA for decision making. In terms of usability, the overall rating of FACETS is 4.74, which reflects the information generated by FACETS is easy to use for decision making.

FACETS is an implementation of the SR model. The performance of FACETS, to some extent, supports our initial expectation on the SR model: cognitive decision support. In FACETS, the decision maker's cognitive constructs (mental models and situation awareness) are computerized and represented as information objects, and used to support the process of seeking relevant knowledge and information (situation retrieval) (Q1 and 2). The presentation of the acquired information is also guided by the decision maker's mental models. In this sense, the computerized information processing process is driven by the decision maker's cognition. The ultimate goal of the cognition-driven information processing process is to support the decision maker's cognitive processes for decision making, such as recalling and examining past experience (Q3), perceiving and understanding situation information (Q4, 5 and 6), developing situation awareness (Q9, 10 and 11), and formulating solutions (Q7) for the current decision situation. In this sense, the argument of cognitive decision support in FACETS is made.

10. Conclusions

The idea of facilitating cognitive decision support in BI systems is built upon a reliable theoretical basis which was established through reviewing relevant literature in the following areas: DSS, business management, and cognitive psychology. A large body of research work has identified that managers'

cognition plays an important role during decision making. Furthermore, in today's business environment, decision making is becoming increasingly ill-structured, dynamic, and full of time pressures, uncertainty and personal stakes. This opens up an opportunity to embrace cognitive psychology in order to design and develop new types of BI systems.

In attempting to facilitate cognitive decision support in BI systems, we proposed a SR model to describe the cognition-driven decision process in BI environments. The SR model describes managers' BI-based decision-making behaviors in a conceptual framework. Compared to the traditional DSS-based decision process model, one of the major characteristics of the SR model is that the manager is not only a user of the computer-based information system, but also a central component of the system as a whole. Managers' SA and mental models are utilized for computer-based information processing, and, at the same time, supported by the computer-based information system. A decision process in the SR model is comprised of the manager's cognitive process as well as the computer-based information processing process. The whole decision process is driven by the manager's cognition. The difference between SR and IR is that the former is decision oriented, while the latter is information oriented. The SR model incorporates a recognition-primed decision (RPD) model to describe the actual process of decision making based on SA and mental models. Thus, a decision cycle consists of situation retrieval and RPD-based decision making. A complete decision process includes a number of decision cycles.

FACETS was designed and developed based on the SR model and the corresponding supporting techniques. The development of FACETS was part of the design research as a whole. FACETS was used as a test bed for evaluation of related techniques. FACETS was also evaluated as a system.

The business domain is the context wherein results of this research are presented. However, neither the SR model nor the developed techniques are domain-specific. They can easily be extended to other application domains, such as government, or medical and education, as long as decision-making tasks involve ill-structured problems, uncertainty, high personal stakes, ill defined goals, and time constraints. Therefore, the model and techniques developed in this research also have implications for other types of decision support systems.

It is also clear that the model and techniques put forward in this research have limitations. In the course of this research, managers' situation awareness and mental models are intensively discussed and incorporated into the SR model. The IS techniques developed according the SR model also focus on processing situation awareness and mental models, and supporting the relevant cognitive processes, such as thinking and situation assessment. However, managers' cognition is by no means concerned with only SA and mental models. A wider variety of cognitive processes and cognitive constructs have been identified by cognitive psychologists. We believe a thorough investigation into human cognition from more aspects will yield a deeper understanding of cognitive decision support and thus lead to better decision process models and corresponding techniques. There also exists much room for the improvement of the algorithms and methods developed in this research. For instance, the determination of local contexts of SA sentences is currently based on a point system, which was created through incorporating field experience of language experts. Although the point system can be adjusted manually, it is essentially static during the run time of the system. Machine learning approaches can be employed to improve the accuracy, adaptability and flexibility of this algorithm. The algorithms for query construction can also be improved in several ways: multiple data sources, complex query conditions and information relevance.

Appendix 1. FACETS Use Case

In an use case, three components of FACETS need to be instantiated: the data warehouse, ontology, and experience base. In our experiment, a fictitious organization called Adventure Works was created.

- **Organization Background**

Adventure Works (AW) is an international company specialized in manufacturing and selling bikes, bike accessories and related clothing. AW has subsidiaries in Australia, Canada, France, Germany, United Kingdom, and United States. AW markets their products via traditional resellers and internet outlets.
A data warehouse called *AWDW* was developed for this company, based on a sample database in SQL Server 2005. In the data warehouse, there are 75 tables including seven fact tables and twenty-two dimension tables. The data stored in the sample database covers a wide variety of business sectors such as product, account, customer, geography, reseller and sales.
We created six cubes: *internet sales, reseller sales, sales orders, finance, exchange rates* and *delivery*.

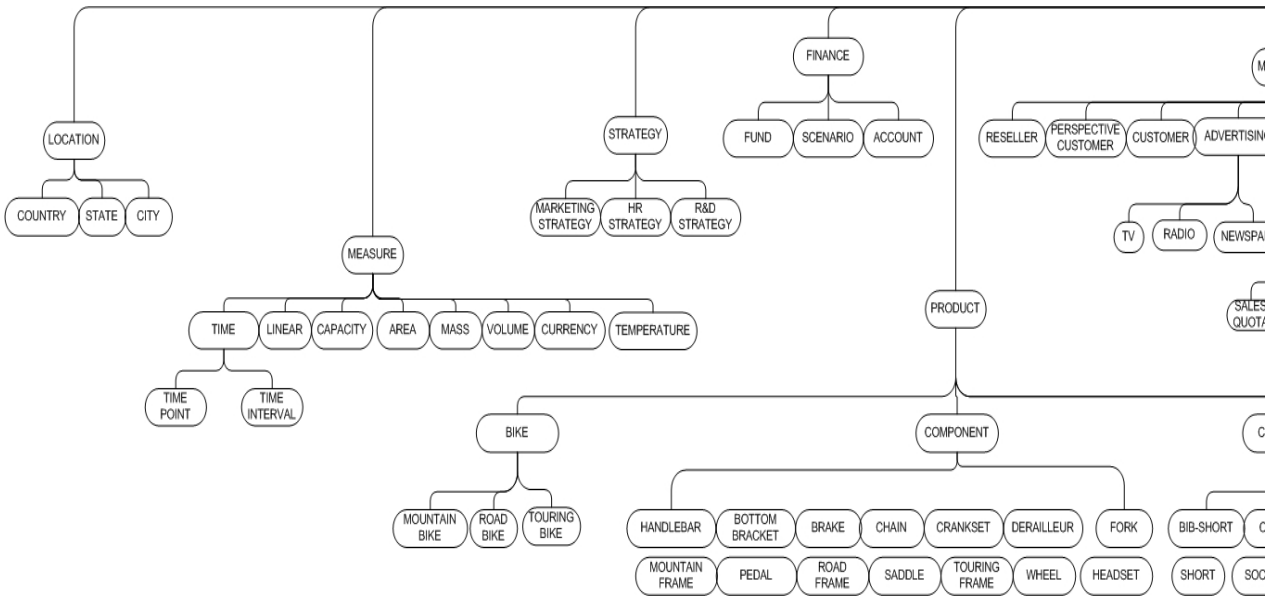


Figure 17 The ClassTree of AWO

- **The Ontology**

According to the technical specification of FACETS, we developed an ontology called Adventure Works Ontology (*AWO*) (Figure 17). In *AWO*, totally 116 classes are defined, corresponding to 115 subsumption relationships in the class tree. Among these classes, 80 classes are correlated to tables or cubes in *AWDW*. For example, class PROMOTION has a correlated table *t_promotion* in the *AWDW*. Class INTERNET SALES has a correlated cube *InternetSales* in the *AWDW*. The excerpt of the class tree of *AWO* is shown in Figure 17.

Over 1000 property-share relationships across the classeswere defined in *AWO*.

- **The Experience Base**

An experience base for FACETS was created. Figure 18 is the excerpt of this experience base.

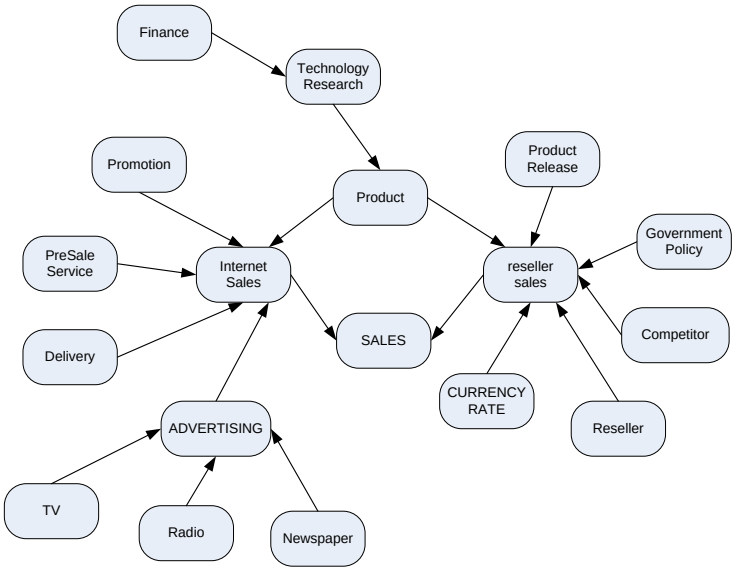


Figure 18 An Excerpt of the Experience Base

- **Decision Situation**

We used a decision situation to illustrate how FACETS could support decision making in the context of AW. This decision situation is described as follows.

AW has been dominating the market for over 10 years. However, a big challenge is coming…
Time: 9:00 AM
Date: 24 December 2003
Mr. Cobarol is the chief executive officer of Adventure Works. Mr. Cobarol has been sleepless for days, because he got a very bad news from the marketing department: the sales of their newly released bike model (BK-M82S-38) have dropped over 40% over the past two weeks. So, with FACETS, how should Mr. Cobarol response appropriately, and reverse the tough situation?

- **Decision Process**

A complete decision process consists of a number of decision cycles. In each decision cycle, the manager inputs into FACETS a description of the current decision situation in form of natural language. The corresponding output of FACETS is the situation information retrieved from the data warehouse according to the situation description. The manager's SA is developed and/or enriched through perceiving the situation information. The improved SA helps the manager to work out a better decision to the current decision situation.

For example, the manager inputs a SA sentence as bellows:

BK-M82B-38 was a very good product.

This sentence represents the manager's initial SA about the current decision situation. It is then parsed, and the final results of SA parsing, which are instances or classes inferred by FACETS from the initial SA, is shown in Figure 19.

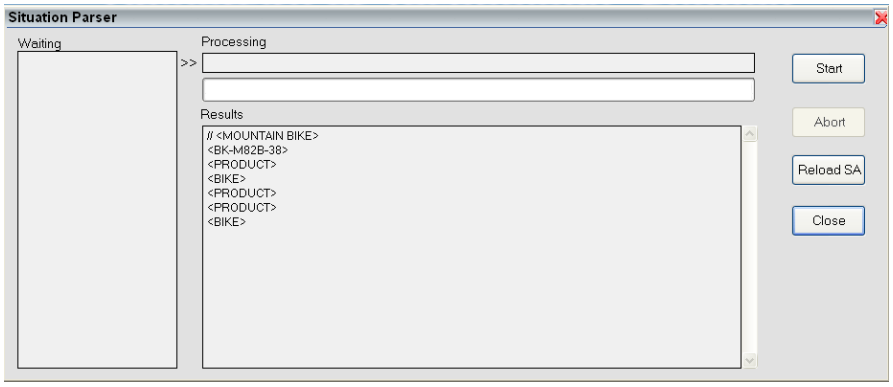


Figure 19 The Results of SA Parsing

The current SA parsing results are then annotated by FACETS, as shown in Figure 20.

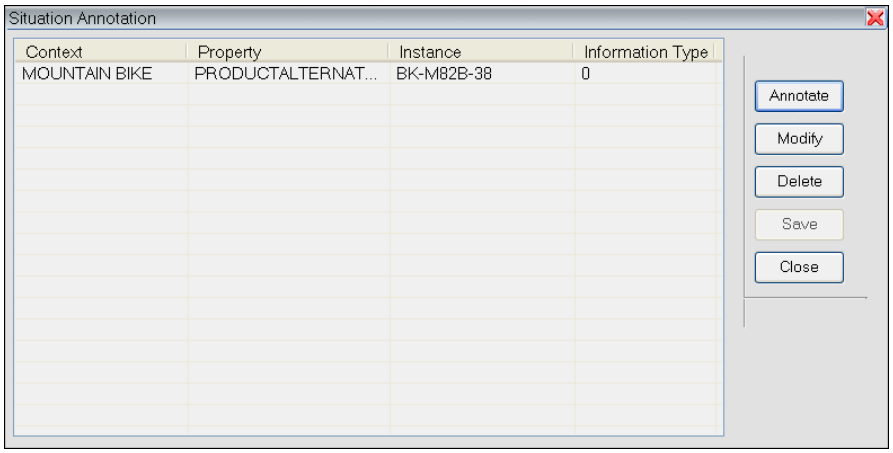


Figure 20 The Results of SA Annotating

The corresponding data warehouse queries are generated by FACETS, as shown in Figure 21.

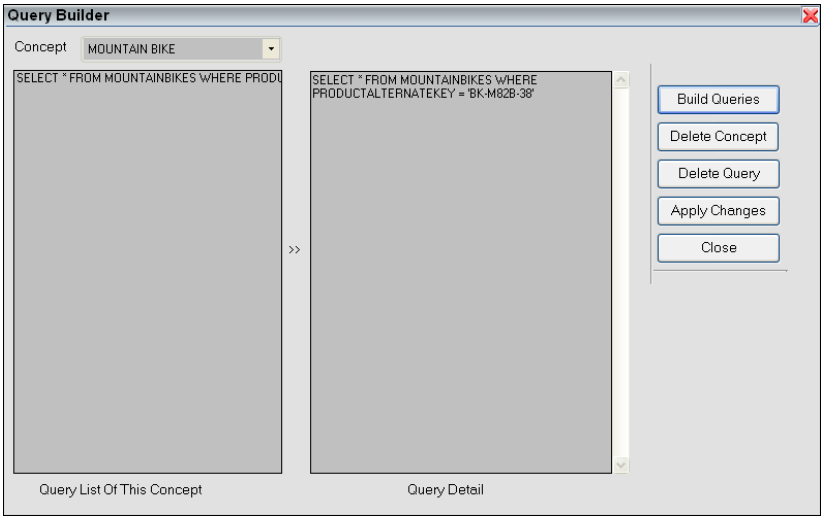
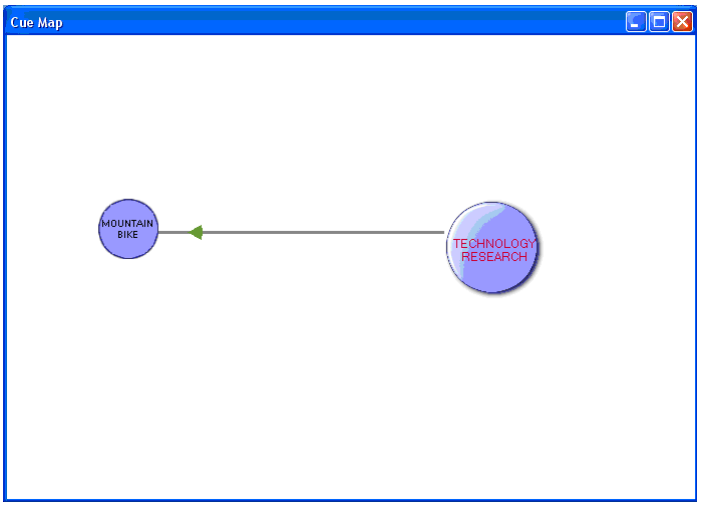
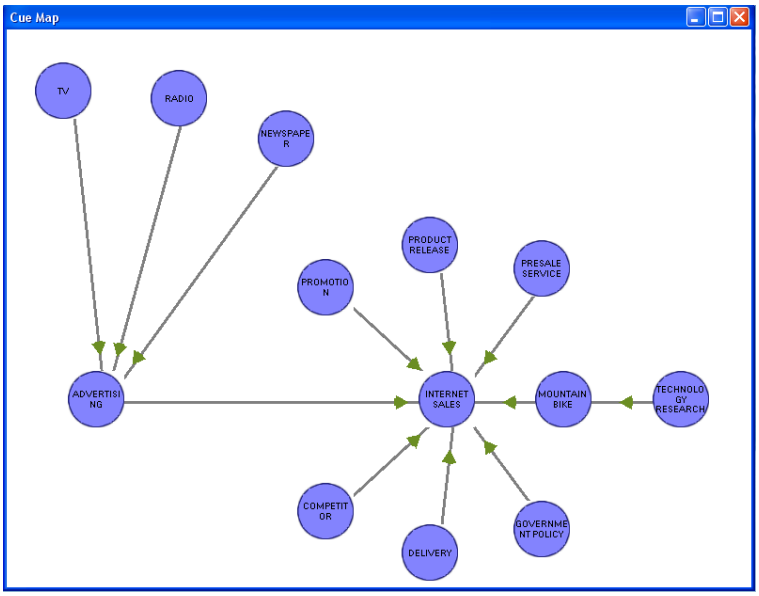


Figure 21 Queries Built by FACETS

The manager then can open the corresponding *Cuemap* (Figure 22).



(a) Decision Cycle 1



(a) Decision Cycle 5

Figure 22 Cuemaps

The cue map is extracted by FACETS from the experience base, according to the manager's input. A node in the cue map represents a concept of the manager's interest in terms of decision making.

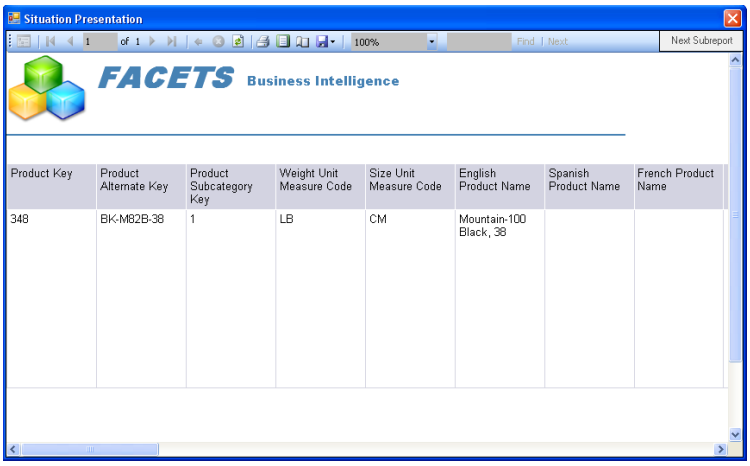


Figure 23 The MOUNTAIN BIKES Report

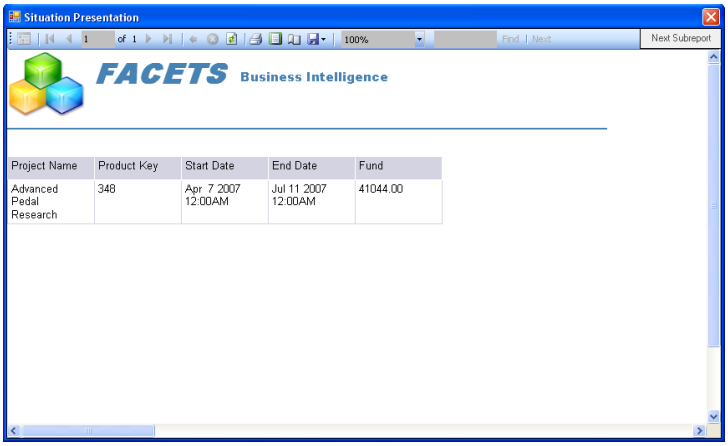


Figure 24 The TECHNOLOGY RESEARCH Report

Clicking on an active concept will open up the report containing information related to current decision situation. The corresponding report of concept MOUNTAIN BIKES and TECHNOLOGY RESEARCH in this decision cycle is shown Figure 23 and Figure 24 respectively.

The manager perceives situation information presented in the reports and develops a deeper understanding of the decision situation. As such, the manager's SA is eventually developed and enriched in this decision cycle.

In the end of decision cycle, if the manager finds himself not confident enough to make a final decision, he can choose to proceed to another decision cycle in order to develop richer SA through acquiring more situation information.

Appendix 2. Glossary

Adventure Works (AW)	A fictitious international company specialized in manufacturing and selling bikes, bike accessories and related clothing.
Causal Relationship	A causal relationship defines an ordered pair of concepts, of which the former concept is a cause of the latter concept.
Class Graph	A class graph is a directed graph representing

	property-share relationships between classes of an ontology.
Class similarity	The class similarity between two classes of an ontology is the numeric measure of how semantically close the two classes are.
Class Tree	A class tree is the tree-like representation of an ontology.
Class Trigger	A class trigger is a term set consisting of a number of terms, which is used as a rule to infer new concepts in semantic parsing.
Context Coverage Point (*CCP*)	The *CCP* points of an alternative context are related to its coverage across the SA sentence.
Context Points (*CP*)	A numeric measure for an alternative context to see if it's a qualified local context.
Context PositionPoint (*CPP*)	The *CPP* points of an alternative context are related to its syntactic position in the SA sentence.
Cue	The cue of a concept includes all factors in an experience which affect this concept no matter whether it is direct or indirect
Experience	Computerized mental models
Experience Map	The representation of an experience which is a directed graph.
FACETS	A cognitive decision support system developed based on situation retrieval model
Graph Similarity	The similarity measure of two classes defined in a class graph
Inverse Context specificity Point (*ICSP*)	The *ICSP* points of an alternative context are related to its relative position in the class tree.
MDX	mdXML is for manipulation of cubes. Many BI vendors support mdXML, like Hyperion, Microsoft, SAP, and SAS. mdXML is based on Microsoft's Multi-dimensional Expressions (MDX), which has been implemented in Microsoft's Analysis Services.
Mental Models	Mental models are commonly referred to as deeply held assumptions and beliefs that enable individuals to make inferences and predictions
Ontology	In computing science, an ontology is a specification of data model about classes (concepts) and their relationships
Plain Parsing	A phrase of situation awareness parsing which recognizes meta instances from situation awareness descriptions.
Precision	the fraction of the relevant documents within the collection of all retrieved documents.
Mroperty-Share Relationships	A property-share relationship between two classes is a

	relationship directed from the intermediary class to the target class.
Recall	the fraction of the documents within the collection of all relevant documents from the data source.
SA Triple	A SA triple has three components: context (*c*), view (*v*), and wording (*w*). The wording part represents a word, phrase, or concept, which is the actual information content the SA triple implies.. The wording of a SA triple is understandable only in a specific context; ambiguity will be introduced without context specification.
Semantic Parsing	A phrase of situation awareness parsing which infers new concepts.
Situation Awareness (SA)	SA is about knowing what is going on around the decision-maker. In business management domain, a manager's SA can be referred to as his/her understanding of the company, e.g., the internal and external environment, the past events, and the current state.
Situation Retrieval (SR)	The process of seeking for information for the purpose of decision making
Structured Query Language (SQL)	The standard interactive and programming language for accessing relational data
Surrogate Class	In a class graph, the surrogate class of a class *c* is the one that has the greatest tree similarity with *c* compared to all other classes of the class graph.
Syntactic Parsing	A phrase of situation awareness parsing based on a syntactic parser.
Taxonomy Property	A special property of a class used to distinguish its child classes.
Tree Similarity	The similarity measure of two classes defined in a class tree.

References

1. Endsley, M.R., B. Bolte, and D.G. Jones, *Designing for situation awareness: An approach to user-centered design*. 2003: London: Taylor&Francis.
2. Singh, S.K., H.J. Watson, and R.T. Watson, *EIS support for the strategic management process.* Decision Support Systems, 2002. **33**(1): p. 71-85.
3. Economist Intelligence Unit, *What do companies want from business intelligence?* 2006, Economist Intelligence Unit.
4. Sutcliffe, K.M. and K. Weber, *The high cost of accurate knowledge.* Engineering Management Review, IEEE, 2003. **31**(3): p. 11-11.

5. Chen, J.Q. and S.M. Lee, *An exploratory cognitive DSS for strategic decision making.* Decision Support Systems, 2003. **36**(2): p. 147-160.
6. March, J.G. and H. Simon, *Organizations.* New York: Wiley, 1963.
7. Donaldson, G. and J.W. Lorsch, *Decision making at the top: The shaping of strategic direction*. 1983: New York: Basic Books
8. Porac, J.F. and H. Thomas, *Taxonomic mental models in competitor definition.* Academy of Management Review, 1990. **15**: p. 224-240.
9. Mintzberg, H., *The nature of managerial work*. 1973: New York: Harper & Row.
10. Isenberg, D.J., *How senior managers think.* Harvard Business Review, 1984: p. 82–90.
11. Schmitt, N., *Naturalistic decision making in business and industrial organizations.* Naturalistic decision making, 1997: p. 91-98.
12. Pohl, R.F., *Cognitive illusions*. 2004: New York: Psychology Press.
13. Russo, E. and P.J.H. Schoemaker, *Decision traps: Ten barriers to brilliant decision-making and how to overcome them*. 1990: New York: Simon And Schuster.
14. Barnes, J.H., *Cognitive biases and their impact on strategic planning.* Strategic Management Journal, 1984. **5**(2): p. 129-137.
15. Yadav, S.B. and D. Khazanchi, *Subjective understanding in strategic decision making: an information systems perspective.* Decision Support Systems, 1992. **8**(1): p. 55-71.
16. Ram, S. and D.A. Carlson, *HyperIntelligence: the next frontier*. Commun. ACM, 1990. **33**(3): p. 311-321.
17. Lee, K.C. and H.S. Kim, *A Fuzzy Cognitive Map-Based Bi-Directional Inference Mechanism: An Application to Stock Investment Analysis.* Intelligent Systems in Accounting, Finance & Management, 1997. **6**(1): p. 41-57.
18. Noh, J.B., et al., *A case-based reasoning approach to cognitive map-driven tacitknowledge management.* Expert Systems with Applications, 2000. **19**(4): p. 249-259.
19. Konar, A. and U.K. Chakraborty, *Reasoning and unsupervised learning in a fuzzy cognitive map.* Information Sciences, 2005. **170**(2–4): p. 419-441.
20. Stylios, C.D., et al., *Fuzzy cognitive map architectures for medical decision support systems.* Applied Soft Computing, 2008. **8**(3): p. 1243-1251.
21. Kuo, F.-Y., *Managerial intuition and the development of executive support systems.* Decision Support Systems, 1998. **24**(2): p. 89-103.
22. Thomassin Singh, D., *Incorporating cognitive aids into decision support systems: the case of the strategy execution process.* Decision Support Systems, 1998. **24**(2): p. 145-163.
23. Lu, J., L. Niu, and G. Zhang, *A Situation Retrieval Model for Cognitive Decision Support in Digital Business Ecosystems.* IEEE Transactions on Industrial Electronics, 2012. **PP**(99): p. 1-1.
24. Ingwersen, P., *Information retrieval interaction.* 1996: London: Taylor Graham
25. Rouse, W.B., *Need to know-information, knowledge, and decision making.* IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 2002. **32**(4): p. 282-292.
26. Guarino, N. *Formal ontology in Information Systems*. in *the 1st International Conference on Formal Ontologies in Information Systems*. 1998. Trento, Italy: IOS Press.
27. Schickel-Zuber, V. and B. Faltings. *OSS: a semantic similarity function based on hierarchical ontologies*. in *the International Joint Conference on Artificial Intelligence 2007*. 2007. Hyderabad, India.
28. Leacock, C. and M. Chodorow, *Combining local context and WordNet similarity for word sense identification*, in *Wordnet: an electronic lexical database*, C. Fellbaum, Editor. 1998, MIT Press: Cambridge. p. 265 – 283.
29. Ackermann, F., S. Cropper, and C. Eden, *Getting Started with cognitive Mapping*, in *7th Young OR Conference, University of Warwick*. 1992: Coventry CV4 7AL, UK.

30. Carlsson, C. and R. Fuller. *Adaptive fuzzy cognitive maps for hyperknowledge representation in strategy formation process*. in *International Panel Conference on Soft and Intelligent Computing*. 1996. Budapest, Hungary.
31. Langfield-Smith, K. and A. Wirth, *Measuring differences between cognitive maps.* The Journal of the Operational Research Society, 1992. **43**(12): p. 1135-1150.
32. Temperley, D. and D. Sleator, *Parsing English with a Link Grammar*, in *Third International Workshop on Parsing Technologies*. 1993.
33. Uschold, M., et al., *The enterprise ontology.* The Knowledge Engineering Review, 1998. **13**: p. 31-89.
34. Clarke, S.J. and P. Willett, *Estimating the recall performance of Web search engines.* Aslib Proceedings, 1997. **49**(7): p. 184-189.