INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
ICED 05 MELBOURNE, AUGUST 15-18, 2005

A FRAMEWORK FOR DESIGN RATIONALE RETRIEVAL

Sanghee Kim, Rob H. Bracewell, and Ken M. Wallace

Keywords: Design rationale capture and retrieval, semantic annotation, question answering

1 Introduction

Design documents contain various types of knowledge and these are useful resources for knowledge reuse. The increasing interest in capturing and reusing such documents is evident as employees in knowledge-intensive areas tend to rely on experience. For example, the majority of engineering designs are related to the modifications of previous proven designs in order to accommodate new requirements. Knowledge reuse depends on the successful retrieval of the required knowledge. It has been difficult to develop efficient retrieval methods especially for the documents created by individuals. When people create documents, they might not consider how the documents can be reused or what information should be accessed. This paper presents a semantic-based retrieval method that understands, organizes and extracts information from documents in a way that mimics human thought (e.g. *what are the causes of overheating?)*. One advantage of this method is that information seekers do not need to worry about selecting right query terms.

2 Background

In engineering companies, where knowledge is increasingly seen as a prime asset, there is great concern about the loss of expertise, as staff retire or move to other companies. However, if engineers could be provided with a suitable tool to capture their rationale routinely and unobtrusively in a form easily understood by others, *and be persuaded to use it*, then the problem would be greatly alleviated. The Design Rationale editor (DRed) is a software tool that was designed to meet this need, supplanting traditional designer's notebooks, as a way of capturing design decisions and their justifications at the time that they are made [1,2]. Unlike previously proposed tools of its type, engineering designers *do* appear to be willing to use it. Its use has been increasing steadily in a leading multinational aerospace manufacturer for the last two years. Rather than being imposed by edict from the top down, it was simply made available to designers who wished to try it, and its usage has spread organically through personal recommendation. It is now being employed in at least five different sites, on both sides of the Atlantic, and the company has imminent plans to extend it to external outsourced design providers. It has been found useful in supporting problem diagnosis as well as design and, and has been used to present the results of design work to external customers. Training requirements are very light. New users generally receive a short introductory hands-on training of 2-3 hours' duration, which feedback has indicated is perfectly adequate. The training was initially given by those responsible for research and implementation of the tool, but recently it has been taken on and further developed by the company's in-house training personnel. A major benefit for designers using DRed as work proceeds, is that the length of the design report required to be prepared at the end of a project is greatly reduced.

Each DRed document, existing as a single conventional computer file, is a scrollable, zoomable plane, onto which textual and graphical elements can be placed and connected by directed links. Figure 1 shows an example snapshot of design work proceeding on the internal gearbox of a turbofan aero-engine. The elements are chosen from a predefined menu of types, at the core of which are issues, answers, pro and con arguments, as proposed in the long established IBIS method for capturing deliberation [9]. Unlike conventional IBIS, every element has graphically displayed status information, updated by the designer as work proceeds. Issue and answer statuses follow a traffic-light convention, while pro and con arguments can be declared dominant, (underlined text), or false (greyed out symbol and strikethrough text). Also shown in Figure 1 are *tunnel links* that provide the means of distributing a single, large rationale over an arbitrary number of small, legible DRed documents. Such links appear to tunnel into the plane and emerge on another, the link continuing to a destination element. Tunnel mouths are represented by small circles and always exist as mutually referencing pairs. Double clicking carries the mouse pointer "through the tunnel" to display the destination document with the far mouth selected. DRed is used in the creation of an on-line design project folder, shared between members of a project team, which may be divided into a tree of sub-folders if desired. The DRed document files are stored in the folder along with those from other software applications used in the design process. These files may be hyper-linked to relevant elements of the rationale, and thereby conveniently accessed.

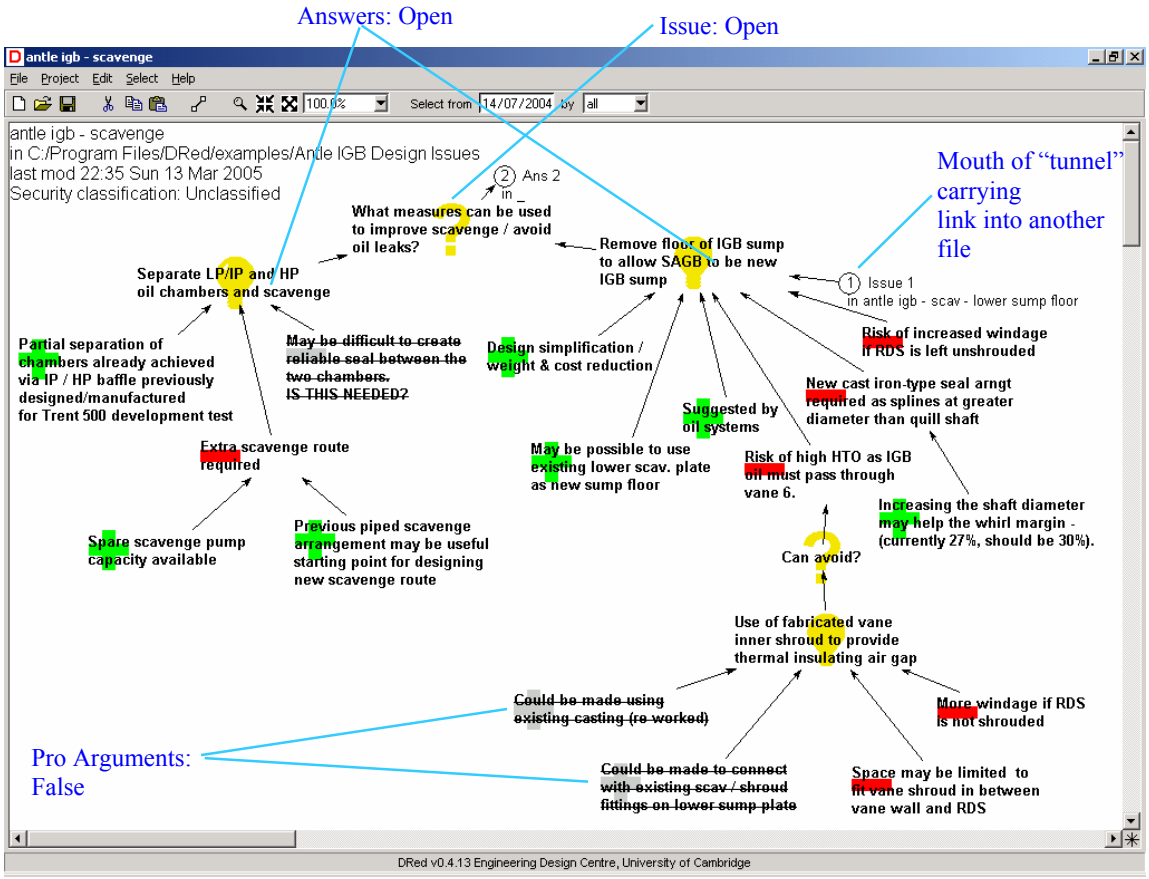

Figure 1. An example of DRed rationale capturing the design of an aero-engine internal gearbox

3 Design rationale retrieval

The majority of design work in the collaborating company involves the modification of previous proven designs whether to accommodate new requirements or to provide better performance by exploiting new materials or technologies. Effective knowledge reuse depends on the successful retrieval of the required knowledge. Unlike most other IBIS derived tools such as Compendium [17], DRed stores its information in a folder of document files rather than using its own dedicated database management system. This undoubtedly eases its introduction into a corporate IT infrastructure but required the creation of an effective way for DRed documents to be retrieved and browsed from a repository of past projects. The approach needed to be compatible with conventional corporate document management systems or intranet web-servers, with DRed used as a helper application along with a standard web browser, configured to handle DRed's .dre file type.

The mechanism works in the following way. On start-up, DRed checks whether another instance of DRed is already running on the same display. If one is, the new instance terminates itself. However, it was invoked with a .dre file to load as a command line argument, it first sends a message to the existing instance to load that file. This mechanism allows a web browser to command DRed to display files that it has downloaded at the user's request. However, the difficulty with this comes when the user wishes to navigate to a linked file by double-clicking a tunnel. DRed does not know the original location of the file downloaded by the web browser, so it has no starting point to follow the relative path and download the linked file. The solution is very simple. The web server is configured by means of a short script, such that when it receives a request for a .dre file, instead of returning the file itself, it returns a file containing just the URL of the file requested. DRed, when passed this file by the browser, is able to download the actual file itself by http, and knowing its location on the server, is able to follow tunnel links to download other files as well. The final point is that DRed needs to be able to tell the server that it wants the actual .dre file, not the file containing its URL. It does this by requesting the file using an http POST request rather than the GET request used by the web browser.

Since .dre files use the plain text gml file format [7] a repository of past design projects can at the most basic be indexed and string searched using any conventional Google-like tool. However the graph nature of DRed documents provides the potential for far more sophisticated approaches to be employed. Conventionally it has been difficult to develop efficient retrieval methods, especially for the documents created by individuals. When people create documents, they might not consider how the documents can be reused or what information should be accessed. People use their own vocabularies and styles defining contents such that the words searchers use are not the same as those by which the information they seek has been indexed. As such, well-defined semantics that provide expressive mark-ups with which meanings and relationships among the texts are essential for efficient retrievals.

3.1 Knowledge reuse through retrieval

Knowledge reuse involves three activities: (1) searching for similar problems or design cases; (2) the recognition of reusable parts of knowledge; and (3) the adaptation of the retrieved knowledge to new requirements. This paper investigates how retrieval methods could better support the first two activities through semantic annotations. Reusability relies on the prerequisite of efficient retrieval, since without knowing what knowledge is available it is impossible to recognise what to reuse. An efficient retrieval method should be able to understand designers' information needs and retrieve answers in a concise and competent way.

Information needs are mainly captured from submitted queries often containing a few terms (i.e. a term is an alpha-numeric expression). Task-involved users (e.g. designers) are known to show similar searching behaviours to non-professional users on the Web, in that both enter only a few terms for queries. Short-length queries tend to be ambiguous and it is hard for retrieval methods to correctly interpret the underlying particular meanings of the keywords contained. Moreover, when the documents are created by individuals, since the words searchers use are not the same as those by which the information they seek has been indexed, it is more difficult to retrieve relevant documents.

When designers have located required knowledge, recognition of which parts of knowledge is reusable takes place. This involves identifying the differences between the retrieved knowledge and the current problems; and exploring implicit assumptions or connected decisions. Designers need motivation to explore, otherwise they might misjudge or fail to recognize appropriate solutions [5]. It is hard to generate pre-compiled reusable knowledge since the relevance of retrieved knowledge depends on the current task context. In particular, design rationale documents present further difficulties as they contain chains of arguments which are only interpretable by domain experts.

Currently, string-based indexing (e.g. a syntactic pattern searching) is the most popular retrieval method. With this indexing, it is hard to recognise that "sufficient" is equivalent to "enough" and to determine that the "blade" as occurs in the following two sentences "*young blade bragged of his amorous adventures"* and *"we need a blade to push against air*" is not identical. It is not able to accept the following two queries "an unfortunate incident" and "a non-serious car crash" as relating to the same topic. It is also difficult to recognise reusable knowledge with the string-based representation, due to its inability to capture sufficient descriptions of given knowledge. Semantics therefore are important for efficient knowledge reuse. Semantics are the meanings of terms and enable to the conversion of given information into easily accessible formats. Natural Language Processing (NLP) is often used for extracting semantics from documents. NLP analyses natural language texts in order to identify what concepts a word or phrase stands for and how to link those concepts together in a meaningful way. With NLP, information seekers are expected to receive more accurate knowledge related to their information needs.

Rationales are the results of complex reasoning and decisions, therefore a simple look-up based retrieval is ineffective [5,6]. A review of current design rationale retrieval research suggests three types of retrievals: (1) a navigation rationales by following links; (2) a query-based retrieval where users submit keywords or select queries from the lists of existing questions; and (3) an automatic trigger that suggests additional information or detects design errors [13]. It is believed that the third type of retrieval meets its needs by interacting with external information systems in order to acquire further information such as task hierarchies or design patterns. User studies have emphasised the need for exploring rationales in relation to implied rationales and the increasing interests in the dependency of information. That is, designers are interested in the dependencies because they want to understand the rationale for the existing design in order to change it [6].

3.2 Premises

In order to understand the issues to be considered when designing an automatic retrieval method to be applied to a DRed repository, an analysis of sample DRed documents was carried out. The documents were collected from engineering designers. The 'what-if' and 'why' types of question were identified as those most relevant to design rationale retrieval. A 'what-if' (i.e. what about x if y?) question is answered with predictions of which parts of

decisions (x) need to be modified when some changes are made (y). 'Why' in general relates to either 'purpose' (i.e. what was the reason for x?) or 'causality' (i.e. what caused x?). In retrieving answers in response to such questions, various types of rationales are required. For example, the dependency information (e.g. "implication") among design parameters is necessary to answer the "what-if" question. Reasoning of how rationales are semantically implied is required. The 'capture-and-replay' approach is hence insufficient as it lacks the capability of dynamically inferring answers from related rationales when the answers are not looked-up. The purpose of this research is to present a framework that models the dependency information among the rationales based on the semantic relations (e.g. 'cause-effect') among them. With this framework, designers do not need to look for additional information, i.e. all related knowledge is automatically extracted and presented to the designers, since otherwise it might be easy for them to misunderstand or fail to judge appropriate solutions.

4 Rationale retrieval framework

The framework is designed to support a retrieval method based on the premises express in 3.2. It aims to augment rationales with well-defined semantics, better enabling computers to understand structured knowledge and perform automated retrieval services. It consists of three discrete components as shown in Figure 2. The first component is a semantic annotation that extracts semantic relations (e.g. 'cause-effect') from the captured rationales. Using the scatter/gathering clustering algorithm [3], similar documents are clustered together to form topics. This is an automatic topic generation method that allows users to browse available topics. A user interface that takes users' queries as inputs, matches the queries with stored semantic annotations, and aims to present corresponding answers in a coherent and competent manner is the third component.

4.1 Semantic annotations using natural language processing

Semantic refers to the meanings of terms and it regards terms as conceptual descriptions of given texts. In comparison to a string-based indexing, using the semantics reduces two well-known problems when dealing with natural language texts. The first is synonymy implying that syntactically different but semantically interchangeable expressions (e.g. overheating vs. excessive heating). Linguistic habits of individuals vary greatly and it was reported that less than 20% of the time two people choose the same keywords for single well-known objects [4]. Synonymy tends to influence "recall" performance. The second is polysemy which means that a word can have multiple meanings. If a retrieval method could reduce ambiguity caused by those two problems, the precision (i.e. the proportion of correctly retrieved documents to the number of documents retrieved) and recall (i.e. the proportion of correctly retrieved documents to the total number of relevant documents) would be greatly improved. One solution is to expand terms with equivalent or related terms. An automatic expansion generates consistent mappings and reduces human's annotation efforts, but a large number of errors could be produced because of "polysemy". For example, "cook" can mean either the person who cooks food or the activity of preparing a hot meal depending on whether it is used as a noun or verb in the text. Since natural language processing is able to analyze terms both from syntactic functions (e.g. noun) and semantics, it provides a better means of collecting the equivalent terms. A collocation (i.e. arbitrary habitual words occurring together like "high temperature") can be also used for disambiguating ambiguous terms. For example, given the "Where can I buy a hot dog?" query, compared to indexing two terms separately (e.g. "hot" and "dog") which involves retrieving any documents that contain either terms, indexing both terms together (e.g. "hot dog") will lead to more precise retrieval. Currently, terms are

expanded with synonyms (e.g. adhesive vs. adherent), variants (e.g. attachment vs. attach) and antonyms (e.g. sufficient vs. insufficient) through referring to the WordNet definitions [10].

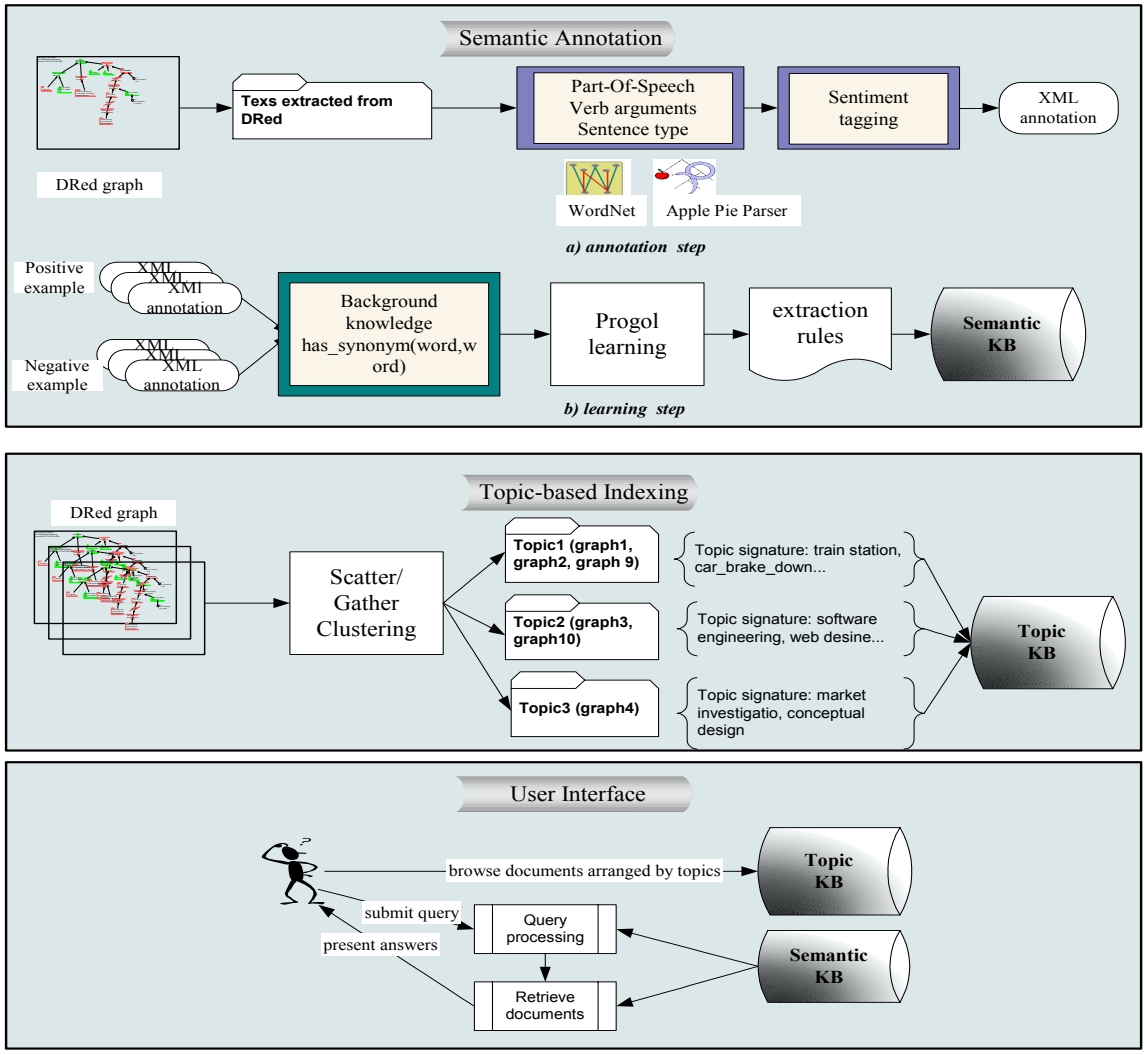

Figure 2. Three components of the rationale retrieval framework

The annotation step in Figure 2 under the component of "Semantic Annotation" shows the procedure of annotations. Each DRed document is first analysed by the syntactic parser [16] for Part-of-Speech taggings and the analysis is recorded in XML files. Morphological analysis (e.g. chances vs chance) is carried out by using stemming and WordNet definitions. Fisher's exact significant test is applied to the XML annotations in order to identify collocations and to differentiate useful collocations from ones that are together by chance. [12]. Each identified verb is parsed with direct and indirect objects.

With well-defined semantics, texts can be analyzed beyond sentence or clause levels to the extent that semantic relationships (e.g. "cause-effect") between two texts can be extracted. The comparisons between two texts then can be made not only by comparing shared terms, but by semantic differences. For example, for the following two sentences, "*Battery is not working (cause). As a result, the car fails to start (effect)",* "cause-effect" relation can be annotated. Sophisticated queries (e.g. what are the causes of "car_break_down?) can be answered with the annotations and it is one of the tasks that current search engines are unable

to handle. With the semantic relations, it is feasible to dynamically compile answers to questions like those in 3.2.

There exist controversies about how to determine the right number of semantic relations and the types. Since too many relation types can increase disagreements among users (i.e. for some users, it may be hard to distinguish 'contrast' and 'concession'), and too few relations may be insufficient for capturing different types of semantics, similar semantic relations are clustered and tried in order to maximise difference among the relations. Semantic relations are included if they are to be used by users when they search and are commonly used across design documents. Table 1 shows selected semantics and examples. Polarity expressions that convey users' opinions on the subject discussed are useful especially to some relations. For example, both for 'motivation' and 'negative-evaluation', the polarities of words can hint 'positive' (e.g. quickest) or 'negative' (e.g. bad) opinions with different degrees (e.g. 'good' in comparison to 'significantly better'). It has been observed that not only adjectives (e.g. good) but also nouns (e.g. failure) and verbs (e.g. disagree) are relevant to determine such polarities. 48 polarity terms were manually constructed and expanded them with synonyms using WordNet.

Table 1. Semantic relations and example texts

Semantic	Description	Example
Cause-effect	Recognising if one state of affairs causes another	*Battery is not working. (cause) As a result, the car fails to start (effect)*
Elaboration	One of two texts contains more details of the another	*Significant improvement in testing (short description). With a new technique, it can diagnosis the problems within seconds (details)*
Evidence	Refers to supporting statements based on experience or similar examples	*There is no petrol in the tank (statment). Fuel gauge shows the tank is empty (evidence).*
Implication	Defines the associated impacts of activities or observations of one state on another state	*If taxi ordered now, it is likely to arrive in time*
Motivation	Positive opinions on ideas or solutions	*Use the car. It is quick (advantage)*
Negative-evaluation	Unfavorable opinions or ideas	*Get a taxi. It is expensive though (negative evaluation)*
Question-Answering	Includes 'confirmation', 'definition', 'quantification' and 'concept completion' questions	*Is this a serious problem? (question), Yes (answer)*
Solutionhood	Presents problems and corresponding solutions	*How to get to the station? (problem) Use the car (solution)*
Walk-around	Around a question or statement which diagnoses given problems	*The car won't start. How to make it start then?*

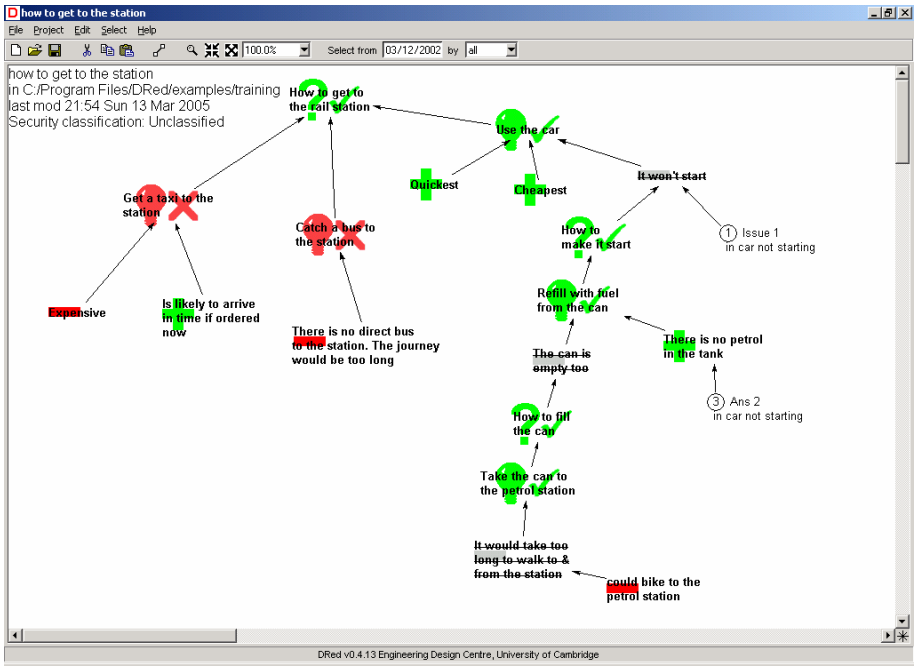

Figure 3. An example of a DRed

Figure 4 shows an example of semantic annotations for the DRed graph shown in Figure 3. With these annotations, users can ask sophisticated queries like '*how to check if there is no petrol in the tank', 'what is the consequence of battery failure?*'. In addition, captured rationale can be retrieved according to the relations, e.g. '*find all causes of car_break_down'*.

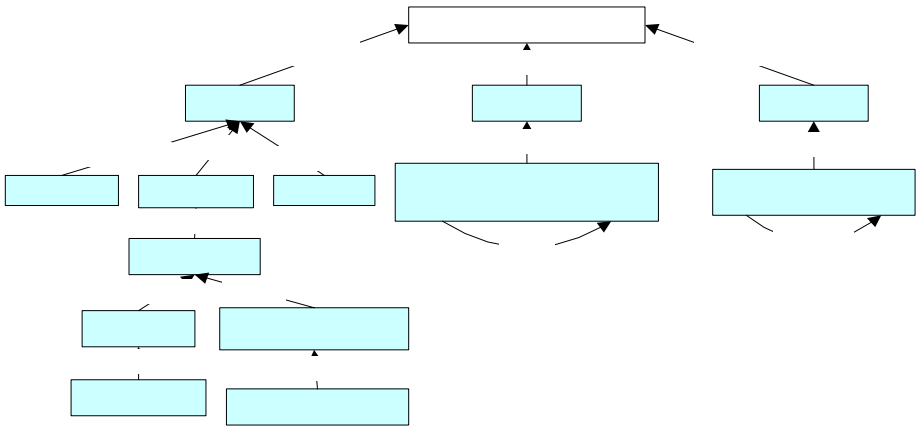

Figure 4. An example of semantic annotations

There have been discussions whether it is more practical to enforce users to provide the semantic annotations rather than automatic annotations. It is known that manual annotations are erroneous and time-consuming tasks often resulting in inconsistent mark-ups. Moreover, user experience with design rationale tools (e.g. IBIS) have revealed that "cognitive overhead" is prevalent and users have difficulty making use of a number of built-in notations when entering their inputs. It means that users might find it difficult to specify their texts with such semantic relations. Automatic annotations are therefore preferred. A supervised machine learning takes DRed documents as training, derives common patterns from the examples and

generates a set of rules that specify how the annotation patterns can be reused for new examples. Progol is an Inductive Logic Programming system that selects one positive example, constructs the most specific clause that becomes a search space for the hypotheses [11]. In Progol, examples are represented with Prolog-style and the generated outputs are relatively easy to understand. Background knowledge is encoded with more flexible formats than attribute-value pairs. A target predicate to be learned is prediction (A, B), where B is the predicted semantic relation with which the link A is to be associated, e.g. prediction(link1, 'contrast'). The learning step in Figure 2 under the component of "Semantic Annotation" shows more details of the automatic semantic annotations. An experiment with 35 design documents showed that the method was effective in obtaining 80% precision when tested with the semantic relations shown in Table 1 [8].

4.2 Topic-based indexing

The second component in the framework is to group semantically close DRed documents together. It is to create a set of topics that designers can navigate through or quickly comprehend a variety of captured rationales without searching. It intends to maximise awareness and provides examples of effective queries by attaching short descriptions to each topic. In the absence of topic maps for engineering design and by considering the time and effort required to create one, an automatic creation of topics from documents is essential. It is also evident that through this approach, the created topics can be kept up-to-date. Clustering algorithms are used for this task. The Scatter/Gather clustering algorithm has been used mainly for assisting users to browse a large collection of documents by allowing them to switch the browsing views from broad to narrow or vice versa [3]. The algorithm is also used for document clustering purposes due to effectiveness in performance: it is a partition method that uses techniques drawn from hierarchical algorithms, but runs in terms of speed, in a comparable way to non-hierarchical methods. Each DRed document is converted into the Okapi term vector model [14] and the similarity between two documents is computed by cosine measurement [15]. For labelling the clusters with terms, a set of keywords that are assumed to specify the central contents of the folder are attached.

4.3 A user interface: Design Rationale Question Answering

This is the application of Question Answering (QA) to design rationale retrievals and allows users to retrieve concise answers to their questions. For example, for the question of "*What are the major causes for aircraft crashes?"* question answering systems retrieve relevant documents and extracts such as "*bombs, mechanical failure, explosive cargos*" as answer fragments. Extracting answers compared with presenting whole documents is beneficial for the users. With a list of documents, the users need to sift through them in order to identify answers to their questions. When the questions are complex and the users have specific information needs, the effort required to find answers increases. Increasing interest in building the Semantic Web is leading to advanced search engines that allow the users to state their information needs in a more specific and natural form. Typical QA systems manually define a table of expected answer types in response to query types. In the proposed approach, the semantic annotations constitute question and corresponding answer types, e.g. "cause-effect". QA applies techniques from natural language and information retrieval to analyse user questions and to extract answers. It is a web-based, platform-independent application and users can access the QA without installing it. To generate dynamic web content, JSP (JavaServer Pages) and Servlet technology from Sun systems have been used. As a backend-storage, MySQL database is used. Given users' submitted queries, QA operates using the following three steps:

1. Query processing: it takes users' queries as inputs and identifies important terms and expands them with equivalent terms. First, a spellchecker examines the terms and corrects commonly known misspellings. Then the query terms are marked either as collocations or single terms. Terms are converted into root formats by using stemming and WordNet definitions. Finally, terms are expanded with equivalent terms.
2. Answer retrieval and scoring: it takes the terms from the step 1 as inputs and searches for answers based on a similarity measurement. Similarity reflects the degree of term correlation, which quantifies the closeness of index terms that occur both in the query and in the answer. It is computed by using the cosine measurement after converting the input texts into Okapi-term vector model.
3. Answer presentation: it takes the similarity values from the step 2 and ranks them in order of relevance. It also highlights terms that occurred both in the query and the answers to help the users to comprehend how the retrieved document is relevant to their queries.

5 An Example

This example demonstrates how the proposed framework can be used to retrieve answers in response to users' queries. It shows how the semantic annotations help users to organise their searches and to retrieve and present the answers. A total of 35 documents was collected from designers in an aerospace company. Figure 5 shows the main screen. Users can type in free text and select one of the available semantics to restrain their query type. Under the heading "Main Topics" in Figure 5, the automatically generated topics (see more details in 4.2) are listed associated with a few terms describing the topics. Users can click each retrieved answer to view a full text. Assume that a newly employed designer considers overheating of oil as a problem in a specific product he is designing. Since it is an abnormal situation, the designer needs to do a diagnosis to identify potential causes for the "heat to oil". Figure 5 shows that the designer enters the query of "heat to oil" and selects "cause-effect" semantic. Figure 6 shows the retrieved results.

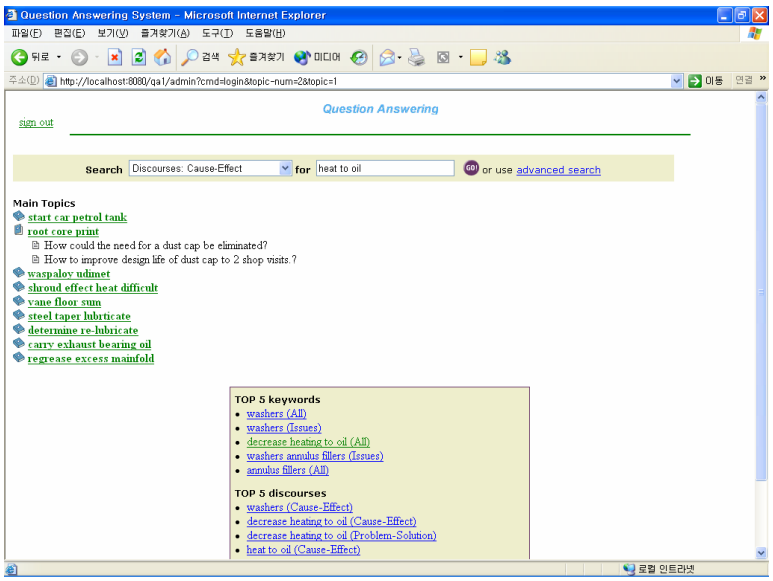

Figure 5. A screenshot of QA's main interface

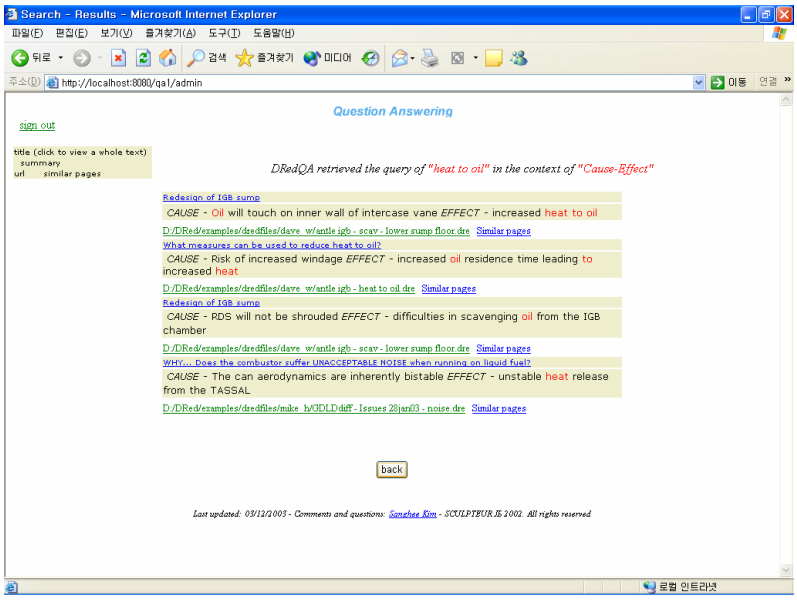

Figure 6. A screenshot of retrieving results within "cause-effect"

The first two answers directly mention the causes of "overheating oil". With the third answer, users have to infer that the problem of oil scavenging can increase heat to oil. Each retrieved answer is shown with the title of the document from which the answer is extracted, a short summary with highlighted terms, and a link to SimilarPages. SimialrPages retrieve similar contents to selected answers irrespective of the query terms submitted. This is a search by "example documents" where users do not have to worry about selecting the correct query terms. After finding out the potential causes of the "overheating oil", the novice designer searches for solutions by selecting the "problem-solution" category and the query contains "heat to oil". Figure 7 shows the results.

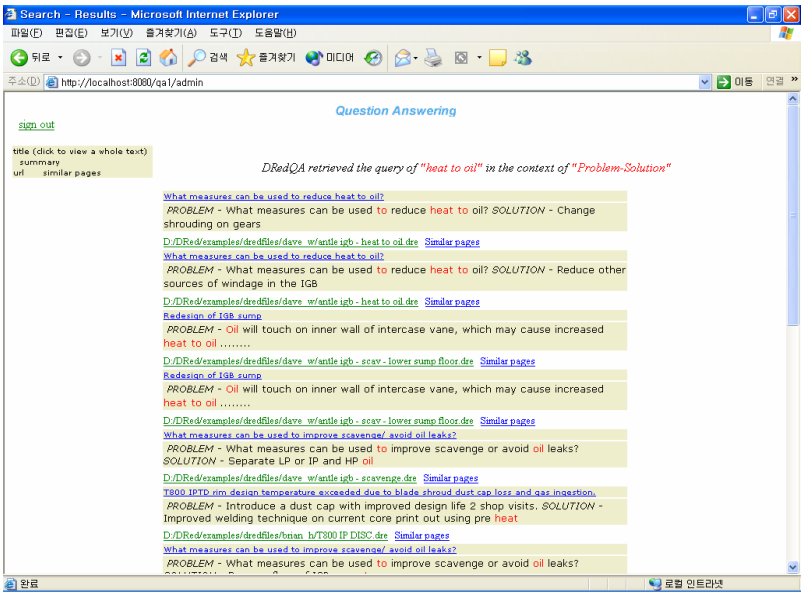

Figure 7. A screenshot of retrieved results by "problem-solution"

From the first and second answer, the designer can recognise that there are two solutions considered by other designers. Figure 8 shows the search results with the same query but without semantic annotations. It is evident that with these retrievals, it is hard to recognise what types of rationales are available for specific problems and users may fail to be aware of the two solutions tried by other designers. Figure 9 shows similar pages searched by selecting the first document shown in Figure 8.

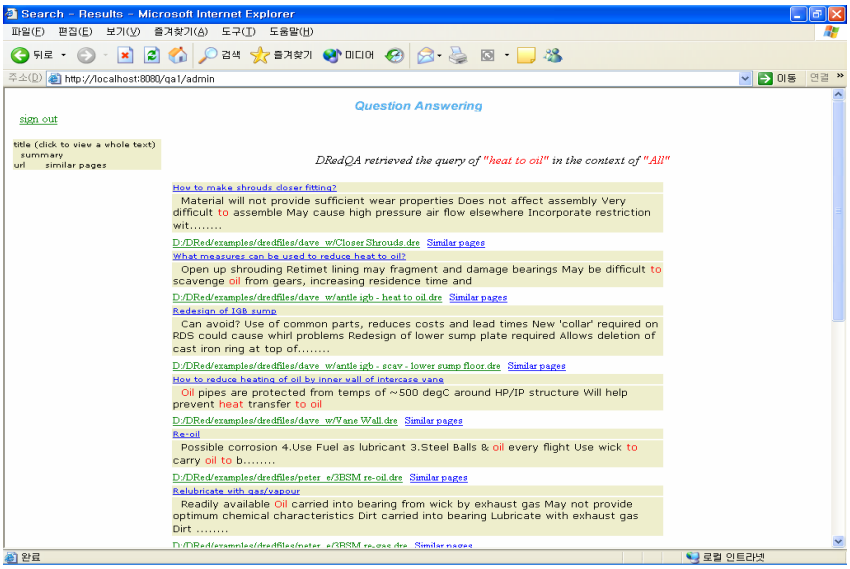

Figure 8. A screenshot of retrieved results by term-matching

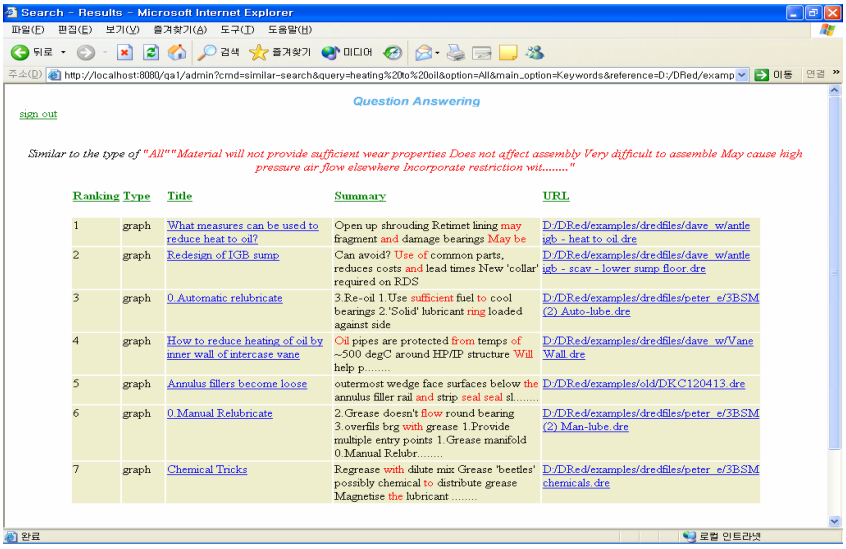

Figure 9. A screenshot for finding similar pages

6 Discussions and Future Work

This paper has presented a design rationale retrieval method based on natural language processing and a supervised learning technique. A main focus is to improve the reuse of

captured rationales through semantic annotations that organise the rationales into well-defined meanings and representations. The semantic annotations help people to know what kinds of rationales (e.g. "cause-effect") are available and to search them with meanings instead of string-matching. In order not to increase users' workloads with regard to annotations, an automatic annotation method has been developed. It allows users to navigate available rationales without searching. Similar documents are clustered into topics and short descriptions of the topics are attached. An example was used to demonstrate the applicability of the approach through graphical user interface. It took only a few minutes to parse and annotate 35 DRed documents with the proposed framework. Another 1-2 hours were spent generating rules using Progol.

In the short term, evaluations for the framework are planned in two directions. First, user evaluations that test whether the answers presented by the proposed approach are easier to recognize compared to the retrievals with a string-based matching. Second, evaluations for the topic-based indexing will be carried out. This will test whether the created topics and the associated short descriptions are meaningful and useful for designers. In the longer term, the application of the proposed approach to other document types is planned. For example, the approach can be used for creating a semantic corporate web since it is able to extract semantics from texts and make use of them for searching. The approach can also be used for ensuring the quality of rationales as it can detect duplicates, contradictions and differences between rationales. For example, it can recognise when new causes to existing problems are captured and compare whether new causes are duplicates or not. Through this comparison, it is feasible to monitor newly captured rationales to identify how these are related to existing ones.

Typical QA systems classify users' submitted queries into expected answer types. For example, given the query of "*where was Bill Gates born?",* the systems search for the parts of texts which contain "location" information. As such, main focus has been on efficient algorithms that understand the queries and identify important terms that highlight expected answers. In comparison to the proposed approach, however, it is infeasible for users to recognise what types of information are available without searching. That is, with QA, users can browse the available knowledge with semantics. With well-defined semantics, it is feasible to automate retrieval services since computers are able to process users' queries autonomously and to generate answers that are not look-up but are inferred from implied rationales.

7 Conclusion

Knowledge reuse depends on the successful retrieval of the required knowledge. Industry experience reveals that employees spend 35% of their time searching for information, while 40% of the corporate users report they cannot find the information they need to do their jobs on their intranets. There exists a huge gap between what a string-based indexing can support and what information seekers expect. When information is abundant, a retrieval method needs to present only timely and relevant information. Semantics are a means of recognising what concepts groups of terms stand for linking those concepts together in a meaningful way for better reuse. This paper presents a semantic-oriented retrieval approach that understands, organizes and extracts information from documents in a way that attempts to mimic human thought. The contribution is summarised as: (1) it enables captured rationales to be retrieved with relation to other implied rationales; (2) it helps people to organise their searches with semantics; and (3) it can dynamically infer answers from implied rationales for users' queries.

8 Acknowledgements

This work was funded by the University Technology Partnership for Design, with industrial partners Rolls-Royce and BAE SYSTEMS.

References

[1] Bracewell, R. H. and Wallace, K. M., "A tool for capturing design rationale", Proceedings of the 14th International Conference on Engineering Design, Stockholm, 2003, 185-186.

[2] Bracewell, R. H., Ahmed, S. and Wallace, K. M., "DRed and design folders: a way of caputuring, storing and passing on - knowledge generated during design projects" in Design Automation Conference, ASME, Salt Lake City, Utah, USA, 2004.

[3] Cutting, D. R., Karger, J. O. and Turkey, J. W., "A Cluster-based Approach to Browsing Large Document Collections", Proceedings of the Fifteen of the ACM Conference on Research and Development in Information Retrieval, 1992, 318-329

[4] Furnas, G. W., Landauer, T. K., Gomez, L. M. and Dumais, S. T., "Statistical semantics: Analysis of the potential performance of key-word information systems", Bell System Technical Journal, 62(6), 1983, 1753-1806.

[5] Garcia, A. C. B. and Souza, C. S., "Add+: Including rhetorical structures in active documents", Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 11, 1997, 109-124.

[6] Gruber, T. and Russell, D. M., "Derivation and use of design rationale information as expressed by designers", Technical Report KSL 92-64, Stanford University, 1992.

[7] Himsolt, M. "GML: A protable Graph File Format", Technical Report, University of Passau, Germany, 1996.

[8] Kim, S., Bracewell, R. H. and Wallace, K. M., "From discourse analysis to answering design questions",Proceedings of the Workshop on the Application of Language and Semantic Technologies to support Knowledge Management Processes, Northamptonshire, U.K., 2004, 43-49.

[9] Kunz, W. and Rittel, W. J., "Issues as Elements of Information Systems", Technical report, 131, University of Berkeley, U.S.A, 1970.

[10] Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D. and Miller, K., "Introduction to wordnet: An on-line lexical database", Technical report, University of Princeton U.S.A.", 1993.

[11] Muggleton, S., "Inverse entailment and Progol", New Generation Computing, 13, 1995, 245-286.

[12] Pedersen, T., "Fishing for Exactness", Proceedings of the South-Central SAS Users Group Conference (SCSUG), Texas, U.S.A., 1996

[13] Regli, W. C., Hu, X., Atwood, M. and Sun, W., "A Survey of Design Rationale Systems: Approaches, Representation, Capture and Retrieval", Engineering with Computers, 16, 2000, 209-235.

[14] Robertson, S. E., Walker, S., Jones, S. and Hancock-Beaulieu, M. G., "Okapi at TREC-3, Proceedings of the Third Text REtreval Conference (TREC-3)", 1995, 550-225.

[15] Salton, G., “Automatic Text Processing, Salton, G. (Ed.), Addison-Wesley Publishing Company, 1989

[16] Sekine, S. and Grishman, R., “A corpus-based probabilistic grammar with only two non-terminals”, Proceedings of the 1st International Workshop on Multimedia annotation, Tokyo, Japen, 2001.

[17] Shum, S. B. and Motta, E. and Domingue, J., “Augmenting Design Deliberation with Compendium: The Case of Collaborative Ontology Design”, Proceedings o the workshop on Facilitating Hypertext-Argumented Collaborative Modelling, ACM Hypertext Conference, 2002

Dr Sanghee Kim
Cambridge University Department of Engineering
Engineering Design Centre
Trumpington Street
Cambridge
CB2 1PZ
United Kingdom
Tel: +44 1223 760559
Fax: +44 1223 332662
E-mail: shk32@eng.cam.ac.uk
URL: http://www-edc.eng.cam.ac.uk/people/shk32.html