INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'09
24 - 27 AUGUST 2009, STANFORD UNIVERSITY, STANFORD, CA, USA

SUPPORTING DESIGN RATIONALE RETRIEVAL FOR DESIGN KNOWLEDGE RE-USE

Hongwei Wang[1], Aylmer L. Johnson[1] and Rob H. Bracewell[1]
(1) Engineering Design Centre, Department of Engineering, University of Cambridge, U.K.

ABSTRACT
Design rationale records the issues addressed, the options considered, and the reasons for making the specific decisions during a design process. However, it is often difficult for a designer to retrieve a particular piece of rationale from a large dataset, which makes the computer-aided retrieval a necessity. Current retrieval methods focus mainly on either the classification of rationale or on key-word based searches of records. The pitfalls of these methods are obvious. First, a classification scheme imposes a large burden on experts. Second, key-word based searches require the designers to understand, at least to some extent, what they are seeking. This ongoing research aims to develop methods and tools for designers to facilitate the re-use of design rationale captured via a software tool currently used in industry. Our first step is to improve the key-word based search by providing better human-computer interaction so that designers can find some useful information even if they cannot explicitly express their information needs. Specifically, the improved retrieval is implemented by a novel combination of three methods, namely suggesting potential key-words to designers, quantifying the relevance of retrieved information, and recommending relevant information based on the dependencies. A software prototype has been developed to demonstrate the proposed approach, as well as to research into how it can be integrated into the design rationale capture tool. Preliminary evaluation shows the prototype can provide good supports to designers especially when they do not know exactly what to search for. Some problems encountered during this work are also identified in the evaluation, leading to a discussion of our future work.

Keywords: Engineering design, knowledge management, design rationale, information retrieval

1 INTRODUCTION
A design process essentially involves a large amount of activities related to solving problems and making decisions. A designer's knowledge of what problem is being solved, how a solution is generated, as well as why such a solution can (or cannot) work, is a valuable intellectual asset of an enterprise, and can be recorded as design rationale [1]. As summarised in [1], design rationale can be defined from a variety of views, e.g. "an explanation of why an artifact, or some part of an artifact, is designed the way it is", "what includes all the background knowledge such as deliberating, reasoning, trade-off, and decision-making in the design process of an artifact - information that can be valuable, even critical, to various people who deal with the artifact". Research into how to capture, store, and re-use design rationale has been undertaken in a wide range of domains, e.g. social science, software engineering [2], engineering design [1], etc.

Design rationale can offer designers useful information about how previous designs evolved and in what context such evolution happened. This information is derived from the expertise of designers, and therefore can be viewed as a source of design knowledge. Meanwhile, this information provides important insights into previous designs for other business processes within an enterprise, e.g. the re-design of a product, provision of service to customers, etc. Arora *et al.* listed several benefits of design rationale [3]. First, it will help in recording the design decisions and the reasoning behind the design, which can be later analyzed to control the overall design and manage the complexity of the design process. Second, it will also have its impact on maintenance, which has been observed to be very much dependent on the design. Third, capturing design rationale and providing a suitable representation scheme will also help in reverse engineering of the design and creating libraries of

design-process artifacts for design re-use and design traceability. Fourth, the rationale can be used to reason about the changes made by the designer based on the knowledge base generated.

A large scope of research on design rationale has been undertaken and published elsewhere, e.g. devising rationale models [4, 5], and developing design rationale capturing systems [6, 7]. A review of research into design rationale for engineering design is given in [1]. Compared with capturing design rationale, its subsequent retrieval has gained less attention. This is partially due to the lack of a unified model to represent design rationale. In addition, re-using design rationale has just started to be studied by researchers. However, the retrieval of design rationale is as important as its capture because re-use is the ultimate goal of capturing and storing knowledge assets. Meanwhile, a number of design rationale capturing tools, e.g. the Design Rationale editor (DRed) developed by researchers in the Engineering Design Centre (EDC) of Cambridge University [8], are beginning to be tested and accepted in industry. Preliminary research shows that structured information (captured using DRed) is easier to interpret than traditional design definition reports [9].

With an increasing number of rationale capturing tools becoming accepted in industry, a large amount of design rationale will now be collected, which makes research into retrieving this information a necessity. Some research work has already been undertaken to understand the information needs of designers, to make classification for captured rationale, as well as to develop computer systems to facilitate retrieval. Currently, keyword-based retrieval becomes increasingly mature as powerful methods from other domains begin to be utilised, e.g. Natural Language Processing (NLP) techniques. Nevertheless, these methods have some drawbacks. First, previous research indicates that not all of the designers can explicitly express their information needs [10]. Second, the manual classification of large amounts of rationale is tedious and error-prone. We therefore propose that a combination of retrieval methods can provide better information support for designers. This paper reports our work in progress on developing computer aids for engineering designers to easily and efficiently retrieve design rationale captured in previous design projects.

2 RELATED WORKS

The motivation of our research is to facilitate the retrieval of design rationale for its re-use in new designs. It is therefore related to studies on knowledge management, design rationale, and design information retrieval. A design process essentially involves many tasks related to generating solutions and making decisions. The increasingly complex nature of modern artifacts entails a huge amount of knowledge to support these tasks. Knowledge Management (KM) in engineering design has been studied for a long while, with two specific focuses, namely the personalisation aspect and the codification aspect [11]. Knowledge has been recognized as an important part of product development systems [12]. A specific application of KM in engineering is Knowledge-Based Engineering (KBE) which embeds domain knowledge in systems providing computer aids for designers. For instance, Susca *et al.* presented a KBE application which was used to perform an automatic calculation and evaluation of the mass properties of racing cars [13]. Marx *et al.* developed a knowledge-based system, which was integrated with numerical analysis codes, to evaluate aircraft structural concept, material and process selections [14].

Along with elaborating a more comprehensive representation of design knowledge, e.g. the function, form, and behavior of artifacts, there is also a need to capture design rationale [12]. Design Rationale System (DRS) is also widely studied and developed to facilitate the tasks of capturing, storing and retrieving design rationale. Design rationale or design intent encompasses a wide range of context surrounding the decisions made during a design process. The capture of design rationale is, therefore, related to how a design process is described. For instance, Brazier *et al.* proposed a Generic Task Model for Design (GTMD) which consisted of three subtasks, such that design rationale could be captured from these subtasks [15]. Ganeshan *et al.* used a design decision-making framework to record design histories. Design intent could then be captured from the two characteristics of the framework, namely the objectives and manipulation of those objectives [16]. As design rationale can be used to illustrate any decision-making tasks, different kinds of design rationale could be generated during a design process. Therefore, a classification of design rationale is useful in the sense that it can

provide insights for the development of DRS. Shipman and McCall suggested seeing design rationale from three points of view: argumentation perspective, documentation perspective, and communication perspective [17]. These perspectives have complementary advantages in facilitating either the capture or the retrieval of design rationale. They recommended an integration of different perspectives and constructed the PHIDIAS system. The representations methods for design rationale are developed in light of the perspective they emphasize; interested readers are referred to the review paper of Regli *et al.* [1].

Information Retrieval (IR) is an interdisciplinary research field which concerns computer science, mathematical science, library science, linguistics, cognitive psychology, and physics. A lot of research work has been done to study the retrieval of design information. For instance, Charlton systematically studied the retrieval of various types of mechanical design information, from textual information to the structured representation of a design [18]. Natural Language Processing (NLP) and data mining techniques were employed in his study. Recently, research on retrieval of 3D CAD models has attracted much attention, which also requires lots of computational support [19]. Similar to product knowledge and process knowledge, design rationale has also been viewed as an important source of design information. Giess *et al.* proposed to identify and link representation elements of product, process and rationale to support design learning [20]. Huet *et al.* identified design rationale as a key output of design review meetings [21]. In the review paper of Regli *et al.*, they summarized three design rationale retrieval methods, namely a navigation approach, a query-based approach, and a hybrid approach [1]. A navigation approach involves permitting designers to explore design rationale by traversing from one node to another via existing links. A query-based approach provides retrieval strategies according to designers' queries. Approaches using automatic triggers can detect or monitor certain conditions according to the design context. Hybrid approaches provide retrieval strategies based on the combination of the above two approaches.

Kim *et al.* proposed two methods to retrieve design rationale captured using the DRed tool. The first approach is query-based, using NLP techniques to evaluate the similarity of rationale entries [22]. The second approach is proposed by analysing the task models of design re-use, recommending supporting the re-use of design rationale within a design process [23]. Two prototype systems were developed to demonstrate these two methods. The first approach essentially involves establishing semantic annotation for DRed files while the second approach tries to statistically anticipate the next design task likely to be performed by designers. These two approaches are shown to be successful in retrieving DRed files with different focuses. However, there still exist drawbacks in these methods. The query-based approach employs comprehensive NLP and machine learning techniques to establish semantic annotation for documents; but it only works when users can submit an explicit query. The second approach seeks to understand the re-using behavior of designers, and provides automatic suggestions for them; but the method may omit some useful information and does not consider the dependencies between DRed elements.

Compared with previous studies, our research has a slightly different focus on helping users to perform their searching tasks. The first step is to enhance the keyword-based search by improving the human-computer interaction for users even when they cannot submit an explicit query. Based on the analysis of related work, we propose a hybrid approach by a novel combination of the methods proposed in [18], [22] and [23]. Specifically three methods are developed, namely suggesting potential keywords to designers, quantifying the relevance of retrieved information, and recommending relevant information based on the dependencies. With such a retrieval system, designers can either use explicit query or select a term (or a set of terms) suggested by the computer. Meanwhile, once a document is browsed by users, they can navigate to other relevant documents using the auto-suggestions made by the computer.

3 METHODOLOGY

The ultimate goal of this ongoing research is to develop an efficient retrieval method for designers who are using DRed in their daily design tasks. Specifically, the method consists of three main parts, namely understanding designers' needs for retrieving and re-using design rationale, elaborating a

strategy to find useful results to match the queries, and categorizing and indexing rationale records to facilitate searching. The method will be evaluated by developing a prototype tool, with the ultimate aim of integrating this into the DRed tool. The first step is to understand how DRed files are structured, to identify the data structure used to represent the rationales in the depository, as well as to augment the keyword-based search by analysing the dependencies between elements. The section will introduce, in detail, the data model of DRed files and the techniques employed to implement the search.

3.1 An introduction to DRed

Observational studies in the aerospace industry produced the result that a large portion of designers' information queries were answered by approaching another person, although the portion was slightly reduced as improved Information Technology (IT) services were applied in the companies studied [24]. The development of DRed was first driven by a prime requirement, as identified in a collaborative research project investigating the capturing, sharing and re-use of design knowledge in the aerospace industry, to provide a tool capable of capturing design rationale. Such a tool was intended to be used by designers as the design proceeds, not just retrospectively [8, 25]. A study of the existing aero engine design reports showed that much of their contents would map naturally onto the Issue-Based Information System (IBIS) structure, so the rationale was modeled as the issues to be solved, the potential solutions to those issues, and the arguments for or against these solutions [25]. The prime advantage of DRed is its simplicity, which allows it not to obstruct a design [25]. DRed is currently being used in a large aerospace company for daily design tasks and is now part of the engineering training provided by this company for its yearly intake of engineering graduates [9]. DRed, used together with other tools such as CAD packages and analysis tools, can largely replace formal design reports and provide a better method for the interpretation of design information [9, 24]. The basic elements in DRed are shown in Fig. 1.

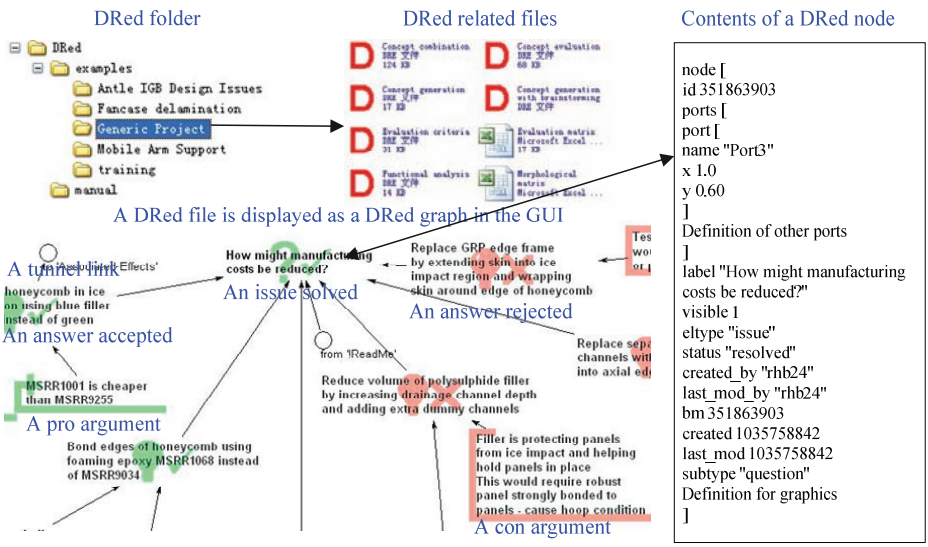

Figure 1. Basic elements in DRed

The rationale captured using DRed are stored as plain text .dre files which can either be saved on a local computer or published via a Web server. A DRed folder is a folder in the filing system of a computer, containing a number of DRed-related files (e.g. spread sheets, DRed files or MS Word documents) needed by DRed to visualize the captured rationale. A large rationale space can be decomposed into a number of smaller pieces each of which is stored as a separate DRed file. Each DRed file can be opened via DRed, and then displayed as a DRed graph (a directed graph of dependencies) in the Graphical User Interface (GUI) of DRed. As shown in the figure, the node of a DRed graph can have a number of types (currently issue, answer, pro-argument and con-argument), as

well as several statuses (accepted, rejected, etc.). A “tunnel link” is introduced to function as an anchor linking two separate DRed files, supporting the navigation of a large graph. The contents of each DRed node are stored in the corresponding DRed file in a structured form; for instance the contents of the node ‘How might manufacturing costs be reduced?’ is shown on the right part of Fig.1. When DRed is started, it loads all the information in a DRed file and displays the graph accordingly.

Originally, DRed was mainly used as a computer aid in the conceptual stage of design, with the focus on supporting the generation of design concepts. It consists of four steps in general to capture a piece of rationale, namely (1) diagnosing the problem; (2) designing a solution; (3) completing a standard checklist template; (4) and communicating the final design and its rationale [24]. Templates are produced to guide designers to perform the necessary capturing tasks, and can be extended if required. Recent extensions to DRed include making bidirectional links between DRed graphs and external files, i.e. spreadsheets, MS Word documents, and CAD files, to enable designers to obtain necessary information from an integrated design space [26]. Such a space can explicitly indicate the correlations between the various sources of information generated during the embodiment design stage, and so allow designers to navigate through, and switch between, these sources.

3.2 An overview of the methodology

An overview of our methodology is shown in Fig. 2, which consists of four main parts. The purpose of retrieving design rationale is to locate relevant pieces of DRed files in response to queries made by designers. In addition, some automatic suggestions can be provided to designers, either before or after the formal queries are submitted. For instance, potential keywords can be suggested to designers and related DRed files, which are not explicitly queried but are clearly needed, can be recommended. All these ideas should be implemented based on the understanding of designers’ information needs, as well as the study of designers’ patterns of information re-use. Once a piece of design rationale is presented to the designers, the goal would be for them to be able to re-use it without any further work. However, this perfect situation may not often occur: in most cases, designers also need to learn how that piece of rationale was adapted in previous designs, by looking through its evolution history. This evolution history is not included in the DRed information model as it can be recorded by storing the evolving files in a conventional versioning repository such as an industrial PLM system. As reading files is expensive in terms of execution time, it’s not reasonable to scan all existing DRed files for each query. Classifying and indexing DRed files can achieve improved search efficiency by storing a summary of the information held in existing DRed files. Furthermore, an efficient searching strategy is also indispensable, not just for achieving a good trade-off between recall and precision, but also for making the best match to designers’ queries. In summary, our ongoing research aims to solve the problems mentioned above. The parts highlighted in Fig.2 are the focus of our current research; the other parts will be studied in our future work.

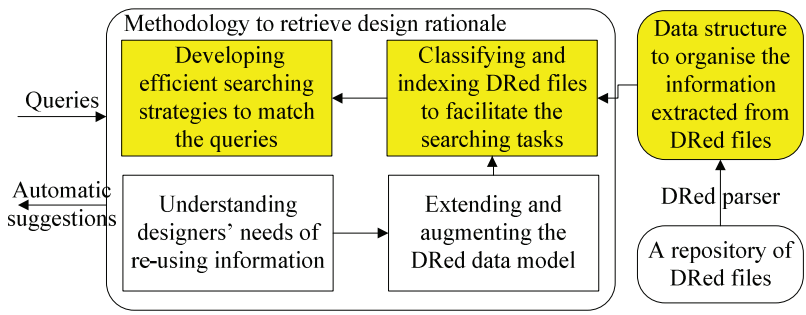

Figure 2. An overview of the methodology

The key to this methodology is the DRed parser which extracts information from DRed files. A data structure also needs to be developed to store the keywords contained in DRed nodes, the contents in DRed nodes, as well as the dependencies between these elements. The extracted information can then be used by a computer aid to facilitate the retrieval. Specifically, the tool can perform the following tasks: (1) suggesting a set of potential keywords for the designers based on the initial letters of the

keywords given by the designers; (2) finding the DRed nodes containing those keywords and ranking the retrieved results; (3) recommending DRed nodes related to the one currently looked at by the designer, based on the similarities and dependencies between the DRed nodes.

3.3 Quantifying correlations and ranking retrieved results

In a DRed graph, the information within each node is mainly represented using plain text, together with pointers to other design information, i.e., spreadsheets, CAD files, analysis programs etc., supplementing the rationale. The retrieval task, therefore, mainly focuses on finding DRed nodes containing a specific keyword, as well as on evaluating the similarity between two nodes. In the following parts of this paper, keywords are called "term" and DRed nodes are called "node", unless otherwise indicated. Generally, there are two criteria for evaluating the performance of a retrieval method, namely recall and precision. The former emphasizes that the retrieval should return as many related results as possible while the latter is used to evaluate the extent to which the returned results will answer the initial query. A number of research questions need to be answered: (1) how to determine the importance of a term within a node; (2) how to deal with synonyms; (3) how to infer the right sense of a term which might have several meanings; (4) how to quantify the contents of a node and evaluate the similarity between any two nodes. Simple Natural Language Processing (NLP) techniques are applied in our research to solve these problems. For question one, the TF-IDF method is used to determine the importance of a term to a node. Let's consider a term t and a node n, the Term Frequency (TF) of t in n can be defined as $tf_{t,n}$ which measures how many times t occurs in n. To make the importance of t increase smoothly as TF increases, a log frequency weight of term t in n can be defined as follows.

$$w_{t,n} = \begin{cases} 1 + \log_{10} tf_{t,n}, & if \ \ tf_{t,n} > 0 \\ 0, & otherwise \end{cases} \tag{1}$$

The premise of TF is that if a term occurs more frequently than others in a node, it tends to be more important to this node as well. Some popular terms, however, often appear in all the nodes, which reduces the accuracy of the TF measure. A measure of the informativeness of a term t is the number of nodes in which it appears. The Inverse Document Frequency (IDF) is proposed to lower to influence of popular terms on their importance to a node, which is defined as follows where N is the total number of nodes and df_t represents the number of nodes in which a term t appears.

$$idf_{t,n} = \begin{cases} \log_{10} N / df_t, & if \ \ df_t > 0 \\ 0, & otherwise \end{cases} \tag{2}$$

Then the importance of a term t to a node n can be determined by the TF-IDF measure as defined in equation (3). Further discussion of the TF-IDF measure can be found in [27].

$$tfidf_{t,n} = w_{t,n} \times idf_{t,n} = (1 + \log_{10} tf_{t,n}) \times \log_{10}(N / df_t), \ tf_{t,n}, df_t > 0 \tag{3}$$

The above TF-IDF measure can be used to quantify the importance of a term to a DRed node. As each DRed node can be characterized by the terms it contains, it can be represented by a vector with values calculated using TF-IDF applied to each term. The similarity between two nodes can therefore be calculated by measuring the "distance" between them. Here, the cosine similarity measure is employed, as shown in equation (4) where n and m are two nodes and V is the set of terms occurring in the two nodes. Thus question four is solved using this similarity measure.

$$\cos(\vec{n}, \vec{m}) = \frac{\vec{n} \bullet \vec{m}}{|\vec{n}||\vec{m}|} = \frac{\sum_{i=1}^{|V|} m_i n_i}{\sqrt{\sum_{i=1}^{|V|} n_i^{\,2}} \sqrt{\sum_{i=1}^{|V|} m_i^{\,2}}} \tag{4}$$

However, the above methods cannot be used to deal with the variations of words and synonyms which are also very important to the performance of a retrieval method. Generally, a word may have a

number of variations, e.g. ‘buy’, ‘buying’, and ‘bought’ can actually mean the same thing when they occur in DRed graphs. In our solution, a database dealing with word variations is used, which increases the amount of data incrementally and can cover most of the common variations. To deal with synonyms, the WordNet definition was used, which provides a large set of words with similar meanings [28]. Although WordNet is not specially designed for the engineering design domain, it proves to be useful in the dataset we used in our simple case study. The last question is about the diversity of the meanings that a term may have. For instance, the word ‘die’ means a special production processing method whilst it may also means losing one’s life as a verb. The precision of retrieval will be affected if this problem is not dealt with properly. A generic approach for such a problem is to narrow down the meaning space of a term by analysing the terms occurring at the same time with it, i.e. the co-occurrence network. This approach also needs an incremental way of expanding the coverage of the network. This approach is left for our future work, and will be developed and integrated into our further development of the prototype system.

3.4 How the DRed parser works

In our research, the retrieval strategy not only aims to find out a set of results in response to a query (submitted as one or more terms) but also puts emphasis on ranking the results and guiding designers to navigate related rationale records. Therefore, our retrieval strategy relies heavily on the correlations built either between terms and nodes, or between nodes and nodes. For instance, the computer needs to find out related DRed nodes for a given query, and also suggest related nodes, based on the content similarity or the dependencies between nodes. The correlation is built by a computer program, the DRed parser, which scans the DRed files through a repository and extracts the useful information to form the data structure for retrieval. The data structure in our study consists of a number lists, namely a “stop word” list (words such as a, an, the, etc. removed during the scanning), a term list, a node list, a synonyms list, a punctuation list (punctuation is commonly used in DRed nodes), and a variation list (storing variations of a single word). With the exception of the term list and node list, these lists simply consist of a set of strings or characters. The data structures of the term and node lists are shown in Fig. 3. A term object has a linked list which contains all the nodes in which this term appears, such that relevant nodes can be rapidly found out once a term is submitted as the query. In addition, a node object generally has three linked lists (the purposes of which are shown in the figure) to enable the computer aid to show nodes which are related to the one currently being looked at by a designer. The algorithms used to extract terms and nodes from DRed files, and to create a list in which to store nodes with similar contents are shown in Fig. 4 and Fig. 5.

3.5 Identifying dependencies between DRed nodes

As discussed above, each DRed file is displayed in the DRed GUI as a directed graph, with tunnel links facilitating the navigation through several files. Any two related nodes in a DRed graph are connected with an arrow indicating the relationship between them, which depends on the types and statuses of the two nodes. For instance, an arrow linking an resolved ‘issue’ node and an accepted ‘answer’ node means that this issue is solved by the answer; an arrow linking an accepted ‘answer’ node and a dominant ‘pro argument’ node means that the answer has been accepted for this reason. In the keyword-based search, only nodes containing the terms of query can be found out as relevant to the retrieval intents of designers. Once a node is retrieved, the nodes connected to it may possibly be useful. In other word, the arrows connecting two nodes in DRed graphs can represent ‘semantic’ relationships which may assist with information retrieval.

In the data model of DRed, there are five types of node, namely ‘issue’, ‘answer’, ‘pro argument’, ‘con argument’ and ‘tunnel link’. Each type is further elaborated using status information, e.g. ‘accepted’, ‘open’, ‘insoluble’, and ‘rejected’ for the ‘issue’ node. In our system, we developed several semantic dependencies by combining these types and statues. It is noteworthy that these dependencies have different levels of importance judged based on the statuses of the two nodes. For instance, an arrow connecting an issue (with the status ‘solved’) and an answer (accepted) is more important than one in which the answer is rejected. Based on the levels of importance, related nodes are recommended with different priorities. With these relationships, designers can navigate a set of nodes without submitting any further queries, as these nodes are automatically suggested by the computer. In such a way, the

proposed approach can have improved precision whilst helping designers to navigate through related nodes which may help them to find more information.

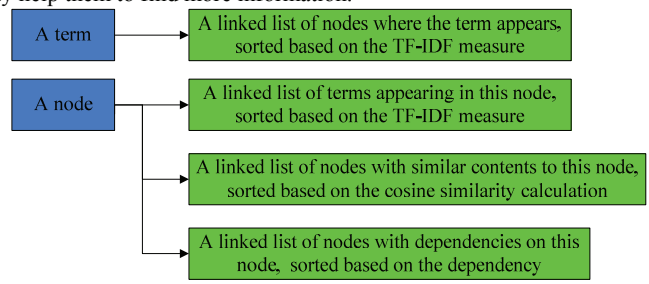

Figure 3. Data structures of term and node

```
ParsingDRedFiles(Dredfile f)
1. Vector termvct=Φ, nodevct=Φ
2. for each line el read from f
3. do judge the type of el
4.    if el represents a node
5.       do create a new node nd and put it to nodevct
6.          for each word wd read from nd
7.             if w already exist in termvct
8.                do increase the TF of wd for nd
9.                   increase the DF of wd
9.             else create a new term tm and put it to termvct
10.   else if el represents a tunnel link
11.         do extract the DRedfile f', do ParsingDRedFiles(f')
12.   if el represents an edge
13.      do create the dependencyList
14. for each node in nodevct
15.   do perform the TF-IDF calculation using equation (3)
```

Figure 4. A recursive algorithm to extract terms and nodes from a DRed file

```
CreatingSimilarNodeList(Vector termvct, nodevct)
1. for each term tm in termvct
2.    for any two nodes in the node list of tm
3.       do get the term list of these two nodes
4.          calculate similarity using cosine method in equation (4)
5.          add each node to the similar-node-list of its counterpart
```

Figure 5. The algorithm to create similar nodes list for each node

4 A PROTOTYPE SYSTEM

To verify the proposed retrieval method, a prototype system is being developed and some preliminary evaluation works have been performed on the current version of the system. Currently, the methods discussed in this paper have been implemented and integrated into the prototype system. The current version of the prototype is now working separately from DRed while in our future work, the functionalities will be integrated into DRed as add-ons. The detailed introduction to the design, implementation and evaluation of the prototype system will be given in this section.

4.1 Framework of the prototype system

As shown in Fig. 6, a three-layer framework has been developed for the prototype system, consisting of a GUI layer, a methodology layer, and a resource layer. Specifically, the GUI layer provides interfaces for enable users' to interact with the computer. In the methodology layer, the methods discussed in previous sections are implemented to receive queries from, and generate data for, the GUI layer. The resource layer comprises the DRed files themselves, as well as the medium to store their indexing and classification. The working procedure of the DRed parser has been discussed above. The DRed parser uses the methodology developed for processing natural language and identifying

dependencies, to generate the information about the terms and nodes extracted from the DRed files repository. The program which bridges the GUI layer and the methodology layer is called a coordinator, and looks after the execution of all the tasks in response to users' operations. For instance, when a user asks the computer to extract information from the DRed files repository, the coordinator will initiate the DRed parser and ask it to scan the repository. Once the scanning is completed, the coordinator will send a message to the user via the GUI.

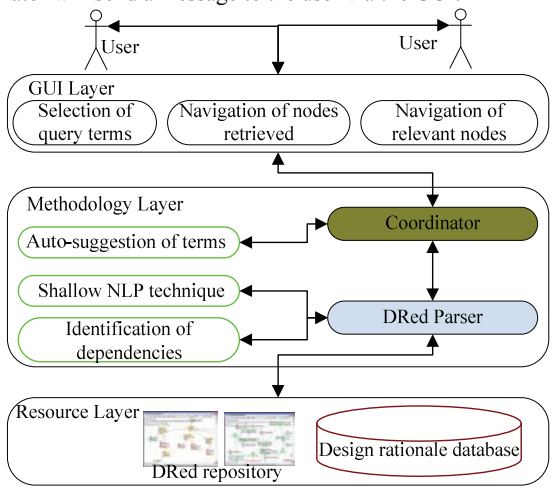

Figure 6. Framework of the prototype system

4.2 Evaluation of functionality

Based on the framework described, a prototype system is being developed and its current version is now being evaluated. The prototype system is being developed as a standalone Java application, whilst the further extension towards a networked version is considered in its design. In the current version, the Database Management System (DBMS) is not used, and information on terms and nodes is simply stored in text files. As the framework introduced in the previous section offers a modular structure, it will not be difficult to transform the prototype system to work together with a DBMS when this becomes necessary. The GUI of the system is shown in Fig. 7, which consists of four main parts. On the left, there is a tree which is used to suggest potential terms (twenty at a time) to the user. With this interface, users can inspect another twenty terms by simply pressing a button. In the central area of the interface, the DRed nodes retrieved are displayed as a list, with their contents shown and keywords highlighted. The right part of the interface aims to enable a user to navigate through DRed nodes that are related to the one currently on display.

The test of functionality involves three tasks, namely using a term suggested, finding out some nodes, and navigating relevant nodes. A simple user study was performed in which two subjects (one of whom has some knowledge about engineering design while the other does not) were asked to use the system. After the user study, both of them said that they liked the method of interaction. They both liked being able to use terms suggested by the computer, as this guaranteed a definite 'result'. To help users to easily find useful terms out of the term list, a filtering mechanism is implemented. A process of filtering the terms using our system is shown in Fig. 8. Two further links come out with each node retrieved, namely a file link and a dependency link. A file link (see for example the highlighted text 'dred\examples\old\DKC166436.dre' under node 1582 in Fig.7) is used to point to the DRed file, which, once pressed, will start DRed to open the file. Each dependency link (see for example the highlighted text 'Dependencies of this document' under the file link in Fig.7) only belongs to a specific node and can trigger the panels on the right part of the GUI to show relevant nodes.

4.3 Evaluation of performance

We evaluated the performance of the prototype system by (1) testing the time to extract information from a repository and (2) comparing the results obtained with previous work done in [22]. Because the data on the terms and the nodes are currently loaded into memory when the system is working, the speed of retrieval is very fast. When the dataset becomes very large, the data will need to be stored in a DBMS, and the speed of the communication with the DBMS may become a bottleneck. We therefore currently just analyse the factors influencing the speed of extracting information. To make the comparison more accurate, we used the same dataset to in the one in [22]. After scanning this dataset, the DRed parser finds 861 DRed nodes and generates 1280 terms based on our methodology. During the benchmarking, the prototype system ran in a PC with 1.83G Hz Duo Core CPU and 2G memory, and completed the scanning within 234 milliseconds. We found that the bottleneck to the speed of scanning DRed files was the operation of reading files on the local drive. Although the utilization of a DBMS might improve this performance, we still believe that further optimisation on the parser might also be possible. We randomly selected a number of terms and tried find out some results in both the prototype system and the system in [22]. A comparison of nodes retrieved is shown in Table 1.

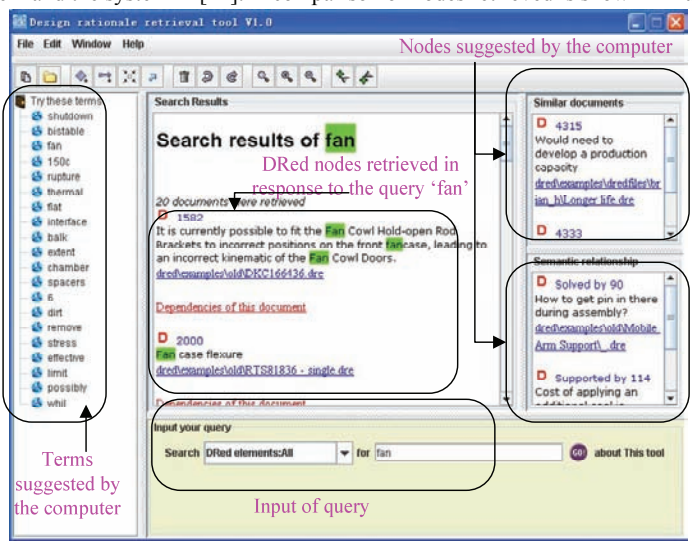

Figure 7. Graphical user interface of the prototype

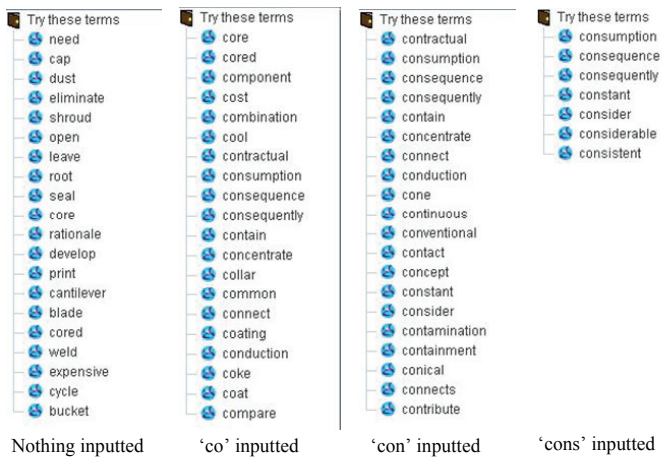

Figure 8. A process of filtering the terms suggested

Table 1. A comparison of results retrieved

Term used	No. of nodes retrieved in the prototype system	No. of nodes retrieved in a previous system [22]
fan	20	10
seal	25	18
cantilever	1	0
autoignition	1	1
blade	17	11
diffusion	2	2
counterbore	1	1

5 CONCLUSION AND FUTURE WORK

We have presented our ongoing research on retrieving the design rationale captured using DRed. The first step has been to improve on simple keyword-based retrieval by enabling better human-computer-interaction. Our methodology involves (1) suggesting terms for users to use; (2) using shallow NLP techniques to quantify the correlations; and (3) providing automatic suggestions for further nodes relevant to the users' current work. A prototype system is being developed and its current version has been used to evaluate our methodology. As evidenced in the test, the basic functionality of the system has been implemented, which is proved to be user-friendly and effective. In addition, the DRed parser can perform the necessary scanning efficiently, and is able to identify most of the information required. Furthermore, a comparison of our prototype with a system developed in earlier research indicates that our prototype can find even more results than the other system, especially for some popular terms. Two reasons can be used to explain this: (1) our system extracts information exhaustively; and (2) the other system may have used some undeclared mechanism for filtering information.

Several issues, however, have also been highlighted in our work to date. First, the parsing algorithm is not yet effective enough, especially the rules to judge whether a term should be extracted. For instance, we can find some numbers extracted by the parser as terms in the current version of the prototype system. Second, our databases for dealing with word variations and synonyms are still not complete, and their coverage of engineering design terms needs to be checked. Third, the retrieval can be affected by the language used by designers who captured the design rationale, e.g. using jargon and special terms. Fourth, our prototype needs further tests and we believe that the running speed can still be optimized. In our future work, we will focus on these issues. Meanwhile, we will continue with the development and study how to integrate the methodology into the DRed tool.

REFERENCES

[1] Regli W.C., Hu X., Atwood M. and Sun W. A survey of design rationale systems: approaches, representations, capture and retrieval. *Engineering with Computers*, 2000, 16(3-4), 269-286.

[2] Jarczyk A. P. J., Loffler P., Shipman III F.M. Design rationale for software engineering: a survey. In *Twenty-Fifth Hawaii International Conference on System Sciences,* 1992.

[3] Arora V., Greer E.J., Tremblay P. A framework for capturing design rationale using granularity hierarchies. In *Fifth International Workshop on Computer-Aided Software Engineering,* 1992.

[4] Lee J. Design rationale systems: understanding the issues. *IEEE Expert*, 1997, 12(3), 78-85.

[5] Klein M. Capturing design rationale in concurrent engineering teams. *IEEE Computer,* 1993, 26(1), 39-47.

[6] Burgess K.C., Conklin E.J. Crisfield M.A. Report on a development project use of an issue-based information system. In *International Conference on Computer Supported Cooperative Work, CSCW 90,* 1990.

[7] Burge J.E., Brown D.C. SEURAT: integrated rationale management. In *International Conference on Software Engineering, ICSE'08,* Leipzig, Germany, 2008.

[8] Bracewell R. H. and Wallace K. M. A tool for capturing design rationale. In *International Conference on Engineering Design, ICED'03,* Stockholm, August 2003, pp.185-186.

[9] Aurisicchio M., Gourtovaia M., Bracewell R.H. and Wallace K.M. Evaluation of how DRED design rationale is interpreted. In *International Conference on Engineering Design, ICED'07,*

Paris, August 2007.
[10] Ahmed S., Wallace K.M. Understanding the knowledge needs of novice designers in the aerospace industry. *Design Studies,* 2004. 25(2), 155-173.
[11] McMahon C., Lowe A., Culley S. Knowledge management in engineering design: personalization and codification. *Journal of Engineering Design,* 2004. 15(4), 307-325.
[12] Szykman S., Sriram R.D., Regli W.C. The role of knowledge in next-generation product development systems. *Journal of Computing and Information Science in Engineering,* 2001. 1(1), 3-11.
[13] Susca L., Mandorli F., Rizzi C., Cugini U. Racing car design using knowledge aided engineering. *Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 2000. 14(3), 235-249.
[14] Marx W.J., Mavris D.N., Schrage D.P. A knowledge-based system integrated with numerical analysis tools for aircraft life-cycle design. *Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 1998. 12(3), 211-229.
[15] Brazier F.M.T., Van Langen P.H.G., Treur J. A compositional approach to modelling design rationale. *Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 1997. 11(2), 125-139.
[16] Ganeshan R., Garrett J., Finger S. A framework for representing design intent. *Design Studies,* 1994. 15(1), 59-84.
[17] Shipman F.M., McCall R.J. Integrating different perspectives on design rationale: Supporting the emergence of design rationale from design communication. *Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 1997. 11(2), 141-154.
[18] Charlton C.T. The retrieval of mechanical design information. *PhD Thesis*, Department of Engineering, University of Cambridge, 1998.
[19] Iyer N., Jayanti S., Lou K., Kalyanaraman Y., Ramani K. Shape-based searching for product lifecycle applications. *Computer-Aided Design,* 2005. 37(13), 1435-1446.
[20] Giess M.D., Goh Y.M., Ding L., and McMahon C.A. Improved product, process and rationale representation and information organisation to support design learning. In *International Conference on Engineering Design, ICED'07,* Paris, August 2007.
[21] Huet G., Curley S.J., McMahon C.A., Fortin C., Sellini F. Communication, information and knowledge processes observed during engineering design reviews. In *International Conference on Engineering Design, ICED'07,* Paris, August 2007.
[22] Kim S., Bracewell R.H., Wallace K.M. A framework for design rationale retrieval. In *International Conference on Engineering Design, ICED'05,* Melbourne, August 2005.
[23] Kim S., Bracewell R.H., Wallace K.M. Improving design reuse using context. In *International Conference on Engineering Design, ICED'07,* Paris, August 2007.
[24] Bracewell R., Wallace K., Moss M., Knott D. Capturing design rationale. *Computer-aided Design,*2009. 41(3), 173-186.
[25] Bracewell R., Ahmed S., Wallace K. DRed and design folders, a way of capturing, storing and passing on, knowledge generated during design projects. In *ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conference, DETC/CIE 2004,*Salt Lake City, USA, October, 2004.
[26] Bracewell R.H., Gourtovaia M., Wallace K.M., Clarkson P.J. Extending design rationale to capture an integrated design information space. In *International Conference on Engineering Design, ICED'07,* Paris, August 2007.
[27] Christopher D., Raghavan P., and Schutze H. *Introduction to Information Retrieval,* 2008 (Cambridge University Press).
[28] The WordNet Project. http://wordnet.princeton.edu/ [June, 2009].

Contact: Hongwei Wang
Engineering Design Centre, Engineering Department, University of Cambridge
Email: hw308@eng.cam.ac.uk URL: http://www-edc.eng.cam.ac.uk/people/hw308.html

Hongwei Wang's research interests are mainly on the methods and tools to support engineering design processes, in particular how to manage design rationale to facilitate knowledge re-use and how to evaluate design concepts using simulation techniques in a distributed environment.