INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'07
28 - 31 AUGUST 2007, CITE DES SCIENCES ET DE L'INDUSTRIE, PARIS, FRANCE

IMPROVING DESIGN REUSE USING CONTEXT

Sanghee Kim[1], Rob H. Bracewell[1], Ken M. Wallace[1]
[1] Engineering Design Centre, Department of Engineering, University of Cambridge, U.K

ABSTRACT
This paper presents an approach of exploiting contexts that are available within design environments in order to improve the understanding of the information needs of a designer. The idea is that such contexts allow reuse tools to acquire information relevant to a designer's current task and this will increase the relevance of retrieved information. A focus is given on the dynamic features of a design process that often mean that a reuse attempt is opportunistic and task-dependent. Current reuse tools have not addressed what types of task models are available and how such models can be used for retrieving existing information. This paper argues that a key element to successful reuse is to interpret the task model as the information seeking behaviours of the designer. To do this, it is necessary to embed the reuse process into design environments in order to closely observe a designer's information seeking processes. The main objective is to reduce the cognitive burden on designers when searching and retrieving reusable information. To demonstrate this idea, a software prototype has been developed and integrated into a design rationale capture tool. Test results indicate that dynamic task models successfully suggest information that not only the designers have not yet anticipated but also that is essential for carrying out their current tasks.

Keywords: design reuse in context, dynamic task model, active retrieval

1 INTRODUCTION
Design reuse aims to maximise the information obtained from previous design activities in order to improve future designs. It involves recognising the need for new information for current designs and retrieving it from design repositories. Given the competitive pressures in business environments, the reuse of previous designs has significant value for shorter delivery times and lower production costs. For example, research has identified that up to 90% of all design activities are based on the variants of existing designs [1]. As such, design reuse can make an important contribution towards improving design efficiency.

Design reuse depends on the results of retrieving the required information. Approximately 90% of organisational memory exists in the form of text-based documents. There has been an increasing interest in retrieving design information from such documents. A keyword-based search is commonly used and whereas it exhibits certain success in facilitating information retrieval, there are two significant limitations when finding the information for reuse: (1) at least 60% of the information that is critical to a designer's task is not accessible [2]; and (2) due to a limited support in recognising the existence of the reusable information, designers often make no attempt at reuse [3, 4].

To address those problems, current approaches have focused on building repository systems that consist of reusable design information. One example is an expert system, which attempts to emulate the problem-solving ability of a domain expert by generating automatic solutions for a specific task. Expert systems have been widely deployed and have led to some significant improvements in knowledge sharing among employees. However, experience of using expert systems highlights the fact that organizational knowledge does not remain static, so a dynamic knowledge acquisition process is needed [5]. Adding new knowledge to existing knowledge bases requires a high level of expert human intervention. It is one of the reasons why the needed information is not always located. In particular, the approaches have not addressed what triggers the designer at the first place to recognise the need of reuse and subsequently initiate a reuse process by formulating searching queries. This means that reuse is often ad-hoc and the designers view the time and effort needed to locate the information and investigate its usefulness as too costly, often resulting in little or no attempt at reuse.

This paper presents an approach for exploiting the contexts available within design environments to gain a better understanding a designer's information needs. The idea is that such contexts are useful to increase the relevance of retrieved information since they provide more precise descriptions of when, how and why a designer perceives a need. It argues that a task model is crucial to identify such reuse intention, and the key to capturing the task is to observe a designer's information seeking behaviour. In this paper, a designer's current task is represented as the sequences of information seeking steps often showing procedure dependencies. To demonstrate this idea, a software prototype has been developed and integrated into a design rationale capture tool. The main objective of this research is to understand how the information captured in previous projects can be made more accessible and usable for future designs. In particular, the investigation and experimentation with a dynamic task model are a primary feature that distinguishes the proposed approach from other research.

2 RELATED WORKS

Previous researchers have investigated the importance of reusing information in practice. Khadilkar and Larry [6] estimated that up to 70% of past design information was requested by designers during redesign. Burge and Brown [7] researched the benefits of reusing design rationales for a large-scale software maintenance task and, through a controlled experiment, they observed that the designers could finish their design tasks in a shorter time. Karsenty also showed the importance of reusing rationales, i.e. over 50% of designer's information needs are related to the questions that could be answered by reusing the rationales [8]. Busby interviewed a total of 50 designers and managers working for two engineering companies in order to identify the causes of reuse failures [9]. A total of 171 failure cases were collected, each of which was analysed according to contributing factors, e.g. organisational reasons. The analysis indicated that the problems were complex and, particularly, the failures could be explained by multiple causes of different types. He suggested that the provision of an easier access to past design information would be helpful. This finding is in line with a new focus within software reuse research that pays more attention to ensuring intelligent storage and retrieval strategies, rather than focusing on developing reusable software components [4].

Focusing on sharing programming codes, e.g. Java API, there are a few well-established development methodologies, e.g. object-oriented design in software engineering. It is claimed that such methodologies can significantly improve software development. In an attempt to investigate whether such methodologies are useful for engineering design, Ruben [10] compared design processes between engineering and software design. He suggested that in software engineering, once the design requirements are satisfied and accepted, they are seldom questioned. On the other hand, engineering design is a selection process involving continuous adjustment and evaluation of the requirements. As such, reuse tools for engineering design need to deal with the dynamically evolving reuse intentions of the designers.

An industry survey reveals that when asked how much of the information that is available to the company is actually used, many organizations responded with a figure of only 20% [11]. This low figure is possibly due to the difficulties associated with locating and retrieving reusable information. Designers are often not motivated to reuse information and can feel overwhelmed by the increasing quantity of information available in a repository. Frequently, the time taken to locate information and the subsequent integration of that information with current new requirements will be perceived as too costly and outweighing any potential benefits. This leads to the main cause of reuse failures, i.e. designers making no attempt to reuse.

Current reuse tools share common shortcomings in addressing the reuse problems above in that: (1) they view a reuse as a standalone process and designers are willing to initiate the reuse process; (2) they assume that the search queries entered by the designers are sufficient for identifying their information needs; and (3) they do not consider that the designer's information seeking behaviour depends on task-dependent procedures. The proposed approach suggests embedding the reuse process into design environments as a better solution to address those problems. Focusing on either Java programming codes or Critics in kitchen designs, CodeBroker, RASCAL, and Critics are most relevant systems to this research [3, 4, 12]. The proposed approach differs from those systems in that it pays special attention to the dynamic nature of the design process, interpreting a designer's current information needs within his or her observed sequence of information seeking behaviours.

3 DESCRIPTION OF THE METHOD

The key element to successful reuse is to understand a designer's reuse intention. The reuse intention is the specification of information needs and is currently normally entered by a designer using a few keywords. It is the first step in a reuse process and unless the designer consciously makes a reuse decision, the process cannot happen. A problem is that the keywords do not provide adequate clues to extract the needed information as represented and stored in reuse repositories. Current approaches to this mismatch problem have focused on supporting advanced query processing, such as partial matching or concept-based indexing. Whereas these approaches can be helpful when the designer makes explicit reuse attempts by submitting such queries, these have limited support when the designer is either not aware of the existence of the information or unwilling to disrupt the design process to search for the information. In fact, this is one of the main reasons for reuse failures and needs particular attention.

An improvement can be made in two directions: (1) actively delivering the information that a designer will need for subsequent tasks so that he or she is aware of available information; and (2) embedding a reuse process within design environments eliminating the need to disrupt the design process to start a reuse query. A core step for such an improvement is to identify what tasks the designer is currently working on and to search for task models that demonstrate similar information seeking behaviour. The reuse tools then use the identified tasks to recommend the next likely information that the designer will need. For example, when faced with an unexpected problem, designers first assess whether or not the problem is serious and requires diagnosis. The designer's intention of reuse is to find reasons or causes that impact on the problem. Once the causes are identified, then it is likely that proven solutions are the next information the designer will need. That is, the information needs can be inferred by knowing what current task the designer is working on since the designer relies on the information to get his or her task done. There are two approaches for identifying the current task of the designer: (1) ask the designer to explicitly define his or her task; and (2) observe his or her design activities.

The first approach assumes that the task model is static and can be inferred approximately from search keywords. Often the model is extended to include the definition of an organisational role, e.g. *a service designer*, or the responsibilities of the role, e.g. *planning product maintenance*. In contrast, this research interprets the task model within a dynamic nature of the design process. The second approach is to use a plan recognition, i.e. a process of inferring plans from observations [13, 14]. It involves a mapping from a *temporal sequence of actions* and their effects to an organization of these actions and their effects into some *plan representation* that identifies the goal of the plan together with the relation between the components of the plan. It has been used for interactive systems such as natural language dialogues where humans interact with computer systems in natural language. In order for computers to understand what goals the humans have, the shared understanding of the humans' tasks is important. Representing users' tasks into a goal-oriented procedure of actions is therefore necessary. In this paper the second approach is adopted and the task model is acquired by observing a designer's design activities.

The proposed approach actively recommends the information that a designer might need to be aware of when carrying out a given design task. The development of the approach consists of two steps: (1) identification of a repository of reusable task models; and (2) construction of the recommendation strategies.

3.1 Task models in DRed

Design Environments are computer-supported programs used by designers exploring problems and finding solutions. As an example, the Design Rationale editor (DRed) is used in this paper. DRed is a software tool that was designed to supplant the traditional designer's notebook as a way of capturing design decisions and their justifications at the time that they are made [15]. The elements are chosen from a predefined menu of types, at the core of which are Issue (I), Answer (A), Pro Argument (PA) and Con argument (CA), as proposed in the long established IBIS method for capturing deliberation [16]. The design rationales captured by DRed are represented as a directed graph where the four elements are linked with one another. The element is further elaborated using status information, e.g. *accepted, open, insoluble* or rejected for the Issue. Each element is associated with a label that is a textual description. Figure 1 shows the links available in DRed. A DRed path is the list of the links starting from a specific element and finishing at a specific element. For example, a DRed path of $I \rightarrow A \rightarrow PA$ can be understood as a supporting argument, i.e. *able to contain the windage from the gears,* is raised for the suggested design option, i.e. *make shrouds closer fitting/fewer gaps*, to the issue created, i.e. *how to change shrouding on gear*. In the context of a design process, the designer has used the DRed path for exploring solutions for a given design task. Such a DRed path is a task model in DRed and the proposed approach recommends the next likely element that the designer will employ.

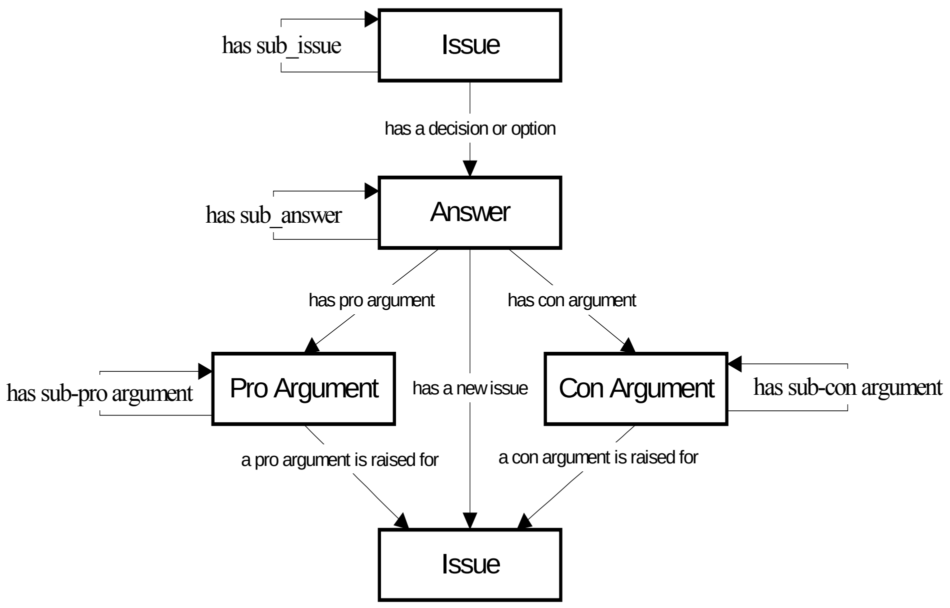


Figure 1. An overview of different link types among the DRed elements

3.2 Recommendation strategies

The proposed approach recommends the element, E_k, using the two strategies: (1) a DRed path similarity; and (2) a content similarity. The first strategy assumes that the DRed paths show how the elements employed in specific sequences are useful for predicting the future elements to be used. That is, the approach examines the sequence in which a current designer has invoked particular elements and uses this as a basis of calculating the prediction for a new element. The following example provides details.

Let us assume that a designer has just created the element, E_l which has a label of L_l, and the approach predicts the next element to be E_k. To do this, an examination of previously captured links (see details in Section 3.1) is carried out and the links of the elements matched with E_l, are assumed as highly relevant. If retrieved links commonly have E_k as the next element of E_l, then, based on the assumption above, it is likely that the current designer would employ the E_k element next. In doing so, the DRed path similarity between the path of the current designer ($|p_i|$) and each of the stored DRed paths ($\overset{n}{\underset{j=1}{}}|p_j|$) is calculated as:

$$sim(p_i, p_j) = \frac{|LP_{ij}|}{|p_i|} * \frac{index_i}{|p_i|} \tag{1}$$

Where, $sim(p_i, p_j)$ measures of correlation between the path of the current designer and one of the stored DRed paths, $index_i$ is the starting position of the correlation, and $|LP_{ij}|$ is the longest path shared between $|p_i|$ and $|p_j|$. Then, the recommended score for E_k is computed as:

$$v(E_k) = \sum_{j=1}^{n} sim(p_i, p_j) * f(k) \tag{2}$$

Where, $v(E_k)$ is the score of the element of E_k to be predicted for a current designer, n is the set of DRed paths stored in the repository, and $f(k)$ is a Boolean indicating whether or not element E_k is invoked directly after the $index_i$ in the path p_j. Equation (1) and (2) are the modifications of the methods proposed in [4] to be used for this approach.

The second strategy compares a content similarity between the label, L_l, and all the labels, $L_{m=1}^{q}$, stored in the DRed repository. In doing so, each captured DRed document is analysed using shallow Natural Language Processing (NLP) techniques. NLP techniques are known to improve the efficiency of document indexing and searches compared to string-based indexing. The techniques include: (1) term identification; (2) Part-of-Speech (POS) tagging; and (3) term normalization. Terms are identified as words lying between two spaces including a full stop. The Apple Pie Parser [17] is used for the POS tagging. POS identifies not what a word is, but how it is used. It is useful to extract the meanings of words since the same word can be used as a verb or a noun in a single sentence or in different sentences. In a traditional grammar, POS classifies a word into eight categories: verb, noun, adjective, adverb, conjunctive, pronoun, preposition and interjection. Each POS-tagged word is compared with WordNet definitions [18] to achieve term normalisation, e.g. *shrouding* is converted into *shroud*. This is to reduce the problem of syntactic variations in grouping similar terms together. Each parsed sentences are then indexed using the Term Frequency Inverse Document Frequency (TFIDF) weighting method [19]. TFIDF assigns numeric weightings to index terms considering the quality of the terms in relation to effective identifiers in the sentences. This allows to distinguish the few sentences in which they occur from the many in which they are absent. The content similarity between the label of the currently invoked element, L_l, and one of the labels in $L_{m=1}^{q}$ is computed using the Cosine Similarity [19], denoted as $c(L_l, L_m)$.

The final recommendation score for the element, E_k, is the summarization of the two strategies:

$$r(E_k) = \alpha * v(E_k) + \beta * c(L_l, L_m) \tag{3}$$

Where, α ($0 \le \alpha \le 1$) and $\beta = (1-\alpha)$ are used to normalize the score to lie between 0 and 1.

3.3. A software prototype

To demonstrate the proposed approach, a software prototype was developed. The prototype aims to: (1) help designers better exploit reuse opportunities by actively recommending useful information which is not known to them and yet might be relevant to their tasks; and (2) reduce the interaction complexity when accessing reuse repositories by eliminating the need for explicitly making reuse queries and for switching contexts between design environments and repositories. It consists of three interfaces: (1) User Profile; (2) Search; and (3) Presentation. Figure 2 shows the architecture. When a designer is capturing and structuring rationales using the DRed, the approach monitors the elements being created and uses this monitored information to recommend the rationales that are deemed useful. When monitoring the designer, only the usage history of current rationales is captured and other rationales that this designer may have previously created are not considered. The User Profile interface represents a current task model as the order of the elements being currently invoked. The prototype starts a background process for capturing and updating the task model continuously without interrupting the designer. Whenever the designer creates a new element and attaches a label to it, the Profile interface sends a query to the Search interface with three parts: (1) the current element; (2) the label of it; and (3) the order of the elements being created. Using the example above, the element being worked on is PA, the label is *able to contain the windage from the gears,* and the DRed path is $I \rightarrow A \rightarrow PA$. The Search interface looks up the existing task models and predicts an element that is likely to be the next element to be used after the PA. For example, the prediction might be the issue related to the PA, e.g., *how much windage can be reduced by putting the shrouds closer.* In order to make such a prediction, it is necessary to construct rules that specify the DRed path from the elements invoked, which represent the *condition* part of the rule, to the predicted element, which represents the *result* part of the rule. Thus, when the current DRed path matches the conditions, the Search interface assumes that the designer is performing the corresponding task and the element in the result part is predicted. The prediction uses the recommendation strategies in Section 3.2. The Presentation interface presents the retrieved elements to the designer by ranking them in decreasing order of relevance to the task model. The prototype was developed using the Perl programming language.

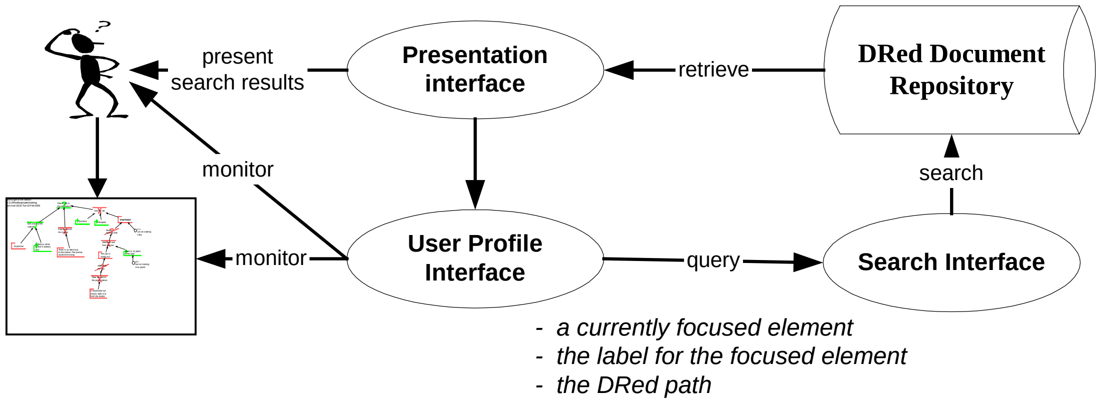


Figure 2. An overview of the prototype.

4 TESTING THE METHODS

4.1 Dataset

This testing uses DRed documents collected from two case studies: (1) one dataset was obtained from a large engineering company, and (2) another dataset was obtained from a project that created DRed documents by reverse engineering documents from the Silent Aircraft Initiative (SAI) research project. An empirical study using the two datasets was carried out in order to identify the usage of elements and link types. Each document was analysed using the prototype in Section 3.3.

The first dataset, referred as *COM*, contains 67 documents and the second dataset, referred as *SAI*, contains 178 documents. These two datasets show different distributions of the elements. In the *COM* dataset, the distribution of each element is: Issue = 21%, Answer = 30%, Con argument = 22% and Pro argument = 17%. In the *SAI* dataset, the distribution of each element type is: Issue = 27%, Answer = 51%, Con argument = 8% and Pro argument = 11%. The distribution of Pro and Con arguments in the *COM* dataset, i.e. 39%, is approximately two times larger than that in the *SAI* dataset, i.e. 19%. The distribution of *Answer* elements with *open* element status in the *COM* dataset is 51% whereas it is 94% in the *SAI* dataset. Both findings confirm that the two datasets are different in terms of their contained DRed document contents. The main design problems in the *SAI* dataset were to investigate various development options for designing a silent aircraft and the exploration of how current design practices could be modified and improved was frequently discussed. The SAI project has currently addressed design issues at the conceptual design stage. It might be difficult to select any design options without considering supporting arguments resulting in the small number of *Answer* element with *accepted* element status in the dataset.

Each DRed document has at least one root element. The root element is the topmost node in a graph at which the rationale capture process begins. Using the links shown in Figure 1, designers can explore different design tasks, e.g. understanding problems or generating solutions. A complete DRed document is viewed as a conceptual model of the design process starting at the root element and finishing at the bottom elements. A total number of DRed paths is 921 for the COM and 352 for the SAI. There exist 278 different types of DRed paths for the COM and 62 for the SAI. Table 1 shows the top five most commonly occurring DRed paths, each of which was associated with the number of occurrences. A single engineer created the DRed documents in the SAI dataset, whereas multiple designers contributed to the 67 documents in the COM. It is one of the reasons why there exist a larger variety of path types in the COM.

Table 1. Top 5 DRed paths in COM and SAI dataset

COM	Num	SAI	Num
Issue(open) → Answer(open) → Con argument(holds)	32	Issue(open)→Answer(open)→ Issue(open)→ Answer(open)	83
Issue(open) → Answer(open) → Pro argument(holds)	29	Issue(open) → Answer(open)→ Pro argument(holds)	56
Issue(resolved) → Issue(resolved) → Answer(accepted)	16	Issue(open) → Answer(open)	30
Issue(open) → Answer(rejected) → Con argument(holds)	15	Issue(open)→ Issue(open)→ Con Argument(holds)	27
Issue(resolved) → Issue(resolved) → Answer(accepted) → Pro argument(holds)	15	Answer(open) → Issue(open) → Answer(open)	15

4.2 Example queries

The prototype uses the recommendation strategies described in Section 3.2. The value for α in Equation (3) was set as 0.6. Two cases of design reuse, i.e. one for each dataset, were prepared. The first case assumes that a designer explores various solutions on the design issue of *reducing heat to oil*. Once he or she identifies that *change shrouding* is one of the options, the designer is looking for design processes that realize the option, and identifies *make shrouds closer fitting/fewer gaps* as one of the options. The left figure in Figure 3 shows the DRed document describing this case and the right figure shows the recommendations presented by the Presentation interface. The corresponding DRed path for the designer's current task is *Issue(open) → Answer(open) → Issue(open) → Answer(open)*. As soon as the label to the currently invoked element, i.e. *Answer(open)* is complete and the designer highlights the element by a mouse-click, the following three pieces of information are sent to the Search interface: (1) *Answer(open)*; (2) *make shrouds closer fitting/fewer gaps*; and (3) *Issue(open) → Answer(open) → Issue(open) → Answer(open)*. The Search interface uses (2) as a query to retrieve the rationales whose labels have similar contents, and (1) and (3) as contexts to rank the retrieved rationales according to their relevance to the designer's current task model. Using the Equation (2), the top three elements that have the highest values are *Issue(open), Con Argument(holds)* and *Pro Argument(holds)*. These are the elements that the designer will most likely use next. For example, the designer might add a negative argument to the current design option, i.e. *would require more shrouding – increases parts count and weight,* or consider the issue of *whether the improved gear shrouding reduces heat to oil problem.* The top ten elements are selected and shown along with the names of DRed documents where each element is extracted. The Presentation Interface allows the retrieved elements to be presented in various ways, e.g. rank the elements as decreasing order of the values in the Equation (3) or as the values in the Equation (2). The right figure in Figure 3 presents the retrieved elements as decreasing values in the Equation (2). Since some of information is regarded as sensitive to the organisation, the texts inside the figure were intentionally made less readable.

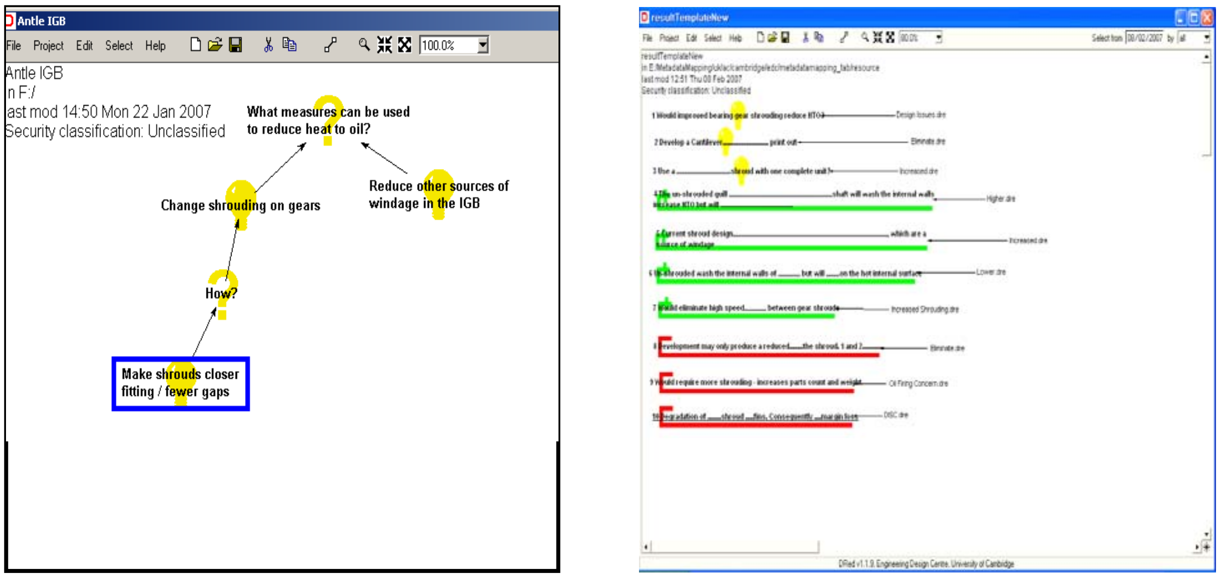
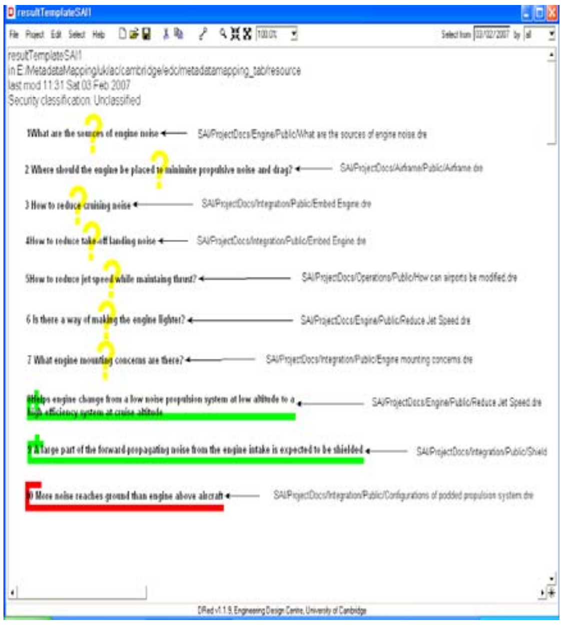

Antle IGB
File Project Edit Select Help
100.0%
Antle IGB
n F:/
ast mod 14:50 Mon 22 Jan 2007
Security classification: Unclassified
What measures can be used to reduce heat to oil?
Change shrouding on gears
Reduce other sources of windage in the IGB
How?
Make shrouds closer fitting / fewer gaps
resultTemplateNew
File Project Edit Select Help
80.0%
resultTemplateNew
in E:/MetadataMapping/uk/ac/cambridge/edc/metadatamapping_lab/resource
last mod 12:51 Thu 08 Feb 2007
Security classification: Unclassified
1 Would improved bearing gear shrouding reduce HTO?
Design Issues.dre
2 Develop a Cantilever____ print out
Eliminate.dre
3 Use a ____ shroud with one complete unit?
Increased.dre
4 The un-shrouded quill ____ shaft will wash the internal walls increase HTO but will ____
Higher.dre
5 Current shroud design____ which are a source of windage
Increased.dre
6 Un-shrouded wash the internal walls of ____ but will ____ on the hot internal surface
Lower.dre
7 Would eliminate high speed____ between gear shrouds
Increased Shrouding.dre
8 Development may only produce a reduced____ the shroud. 1 and 2____
Eliminate.dre
9 Would require more shrouding - increases parts count and weight
Oil Firing Concern.dre
10 Degradation of ____ shroud ____ fins. Consequently ____ margin loss
DISC.dre
DRed v1.1.9, Engineering Design Centre, University of Cambridge

Figure 3. Recommendation results for the first case

The second case assumes that a designer has added a design option of *modify engine to reduce noise* to the issue of *how can the engine be improved*. The corresponding DRed path is *Issue(open) → Answer(open)*. The three pieces of information sent to the Search interface are: (1) *Answer(open)*; (2) *modify engine to reduce noise*; and (3) *Issue(open) → Answer(open)*. Using the Equation (2), the top three elements are *Issue(open), Pro Argument(holds),* and *Con Argument(holds).* The left figure in Figure 4 shows the information suggested using the proposed approach. The right figure shows the information retrieved based on only considering content similarities. That is, in the Equation (3), the value of the DRed path similarity computed by using the Equation (2) was ignored. The first recommended element using the proposed approach is the issue of *what are the sources of engine noise?* This issue is highly useful for the designer since before exploring various solutions to the design option, he or she might want to understand the options better. For example, the information about *the sources of engine noise* is a good starting point to explore *the design process of modifying the engine for noise reduction*. That is, the proposed approach is able to make recommendations in the right context thus helping the designer reuse existing information more effectively. On the other hand, the recommendations in the right figure are the simple look-up results of the rationales whose labels have similar content meaning, so some of them might be irrelevant. Designers then have to spend considerable time examining which presented rationales are most likely to contain their information needs. Using the current task of the designer as a context, the proposed approach demonstrates a more efficient way of retrieving and presenting information.



resultTemplateSAI1
in E:/MetadataMapping/uk/ac/cambridge/edc/metadatamapping_tab/resource
last mod 11:31 Sat 03 Feb 2007
Security classification: Unclassified

1What are the sources of engine noise ← SAI/ProjectDocs/Engine/Public/What are the sources of engine noise.dre

2 Where should the engine be placed to minimise propulsive noise and drag? ← SAI/ProjectDocs/Airframe/Public/Airframe.dre

3 How to reduce cruising noise ← SAI/ProjectDocs/Integration/Public/Embed Engine.dre

4How to reduce take-off landing noise ← SAI/ProjectDocs/Integration/Public/Embed Engine.dre

5How to reduce jet speed while maintaing thrust? ← SAI/ProjectDocs/Operations/Public/How can airports be modified.dre

6 Is there a way of making the engine lighter? ← SAI/ProjectDocs/Engine/Public/Reduce Jet Speed.dre

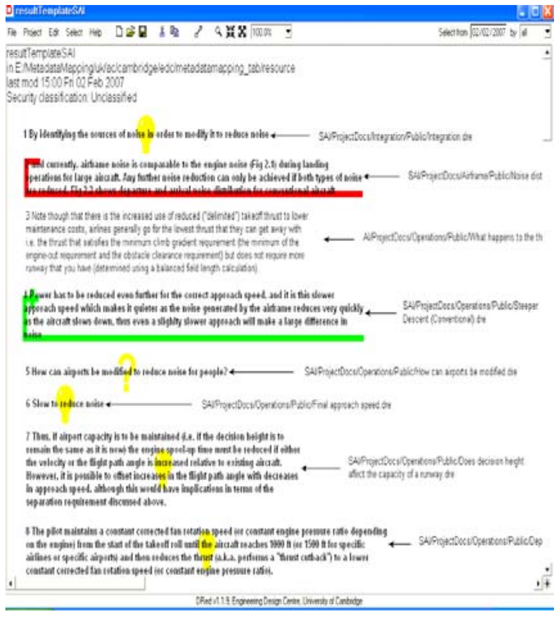
7 What engine mounting concerns are there? ← SAI/ProjectDocs/Integration/Public/Engine mounting concerns.dre

8Helps engine change from a low noise propulsion system at low altitude to a high efficiency system at cruise altitude ← SAI/ProjectDocs/Engine/Public/Reduce Jet Speed.dre

9 A large part of the forward propagating noise from the engine intake is expected to be shielded ← SAI/ProjectDocs/Integration/Public/Shield

10 More noise reaches ground than engine above aircraft ← SAI/ProjectDocs/Integration/Public/Configurations of podded propulsion system.dre

DRed v1.1.9, Engineering Design Centre, University of Cambridge



resultTemplateSAI
in E:/MetadataMapping/uk/ac/cambridge/edc/metadatamapping_tab/resource
last mod 15:00 Fri 02 Feb 2007
Security classification: Unclassified

1 By identifying the sources of noise in order to modify it to reduce noise ← SAI/ProjectDocs/Integration/Public/Integration.dre

2 and currently, airframe noise is comparable to the engine noise (Fig 2.1) during landing operations for large aircraft. Any further noise reduction can only be achieved if both types of noise are reduced. Fig 2.2 shows departure and arrival noise distribution for conventional aircraft ← SAI/ProjectDocs/Airframe/Public/Noise dist

3 Note though that there is the increased use of reduced ("delimited") takeoff thrust to lower maintenance costs, airlines generally go for the lowest thrust that they can get away with i.e. the thrust that satisfies the minimum climb gradient requirement (the minimum of the engine-out requirement and the obstacle clearance requirement) but does not require more runway that you have (determined using a balanced field length calculation) ← A/ProjectDocs/Operations/Public/What happens to the th

4 Power has to be reduced even further for the correct approach speed, and it is this slower approach speed which makes it quieter as the noise generated by the airframe reduces very quickly as the aircraft slows down, thus even a slightly slower approach will make a large difference in noise ← SAI/ProjectDocs/Operations/Public/Steeper Descent (Conventional).dre

5 How can airports be modified to reduce noise for people? ← SAI/ProjectDocs/Operations/Public/How can airports be modified.dre

6 Slow to reduce noise ← SAI/ProjectDocs/Operations/Public/Final approach speed.dre

7 Thus, if airport capacity is to be maintained (i.e. if the decision height is to remain the same as it is now) the engine spool-up time must be reduced if either the velocity or the flight path angle is increased relative to existing aircraft. However, it is possible to offset increases in the flight path angle with decreases in approach speed, although this would have implications in terms of the separation requirement discussed above. ← SAI/ProjectDocs/Operations/Public/Does decision height affect the capacity of a runway.dre

8 The pilot maintains a constant corrected fan rotation speed (or constant engine pressure ratio depending on the engine) from the start of the takeoff roll until the aircraft reaches 1000 ft (or 1500 ft for specific airlines or specific airports) and then reduces the thrust (a.k.a. performs a "thrust cutback") to a lower constant corrected fan rotation speed (or constant engine pressure ratio). ← SAI/ProjectDocs/Operations/Public/Dep

DRed v1.1.9, Engineering Design Centre, University of Cambridge

Figure 4. Recommendation results for the second case

5. CONCLUSIONS AND FUTURE WORK

Current reuse support tools have not efficiently addressed the problem of reuse failures caused by either a designer not being aware of the existence of reusable information or a designer perceiving a reuse process as too costly, thus making no attempt at reuse. A major disadvantage of current tools is that they depend on designers initiating the reuse process. Unless the designer makes an explicit effort at reuse by submitting search queries, design reuse does not happen. Many designers are unwilling to disrupt the design process to search for information. Embedding the reuse process into design environments eliminates such dependency and could improve design reuse significantly. It would also help to exploit task-related contexts for more accurately identifying the information needs and increase the relevance of the retrieved information. The proposed approach has demonstrated the benefits of using a designer's current task model as an example of context for recommending information that the designer has not anticipated and that is essential for implementing the current task. A preliminary test with two reuse queries has shown the potential of the proposed approach.

The proposed approach has currently focused on the design rationales captured by DRed, but it can be easily extended to include other types of resources, e.g. CAD drawing databases. One of the reasons is that the approach has constructed reusable task models with a minimum support from human experts who help identify which tasks are reusable. The user profile can be improved by reflecting the feedback from a designer. Currently, the designer is not allowed to explicitly open his or her profile to customise the content according to his or her preferences. The preferences can be used for determining how many details of retrieved information should be shown since a concise and short information might be preferred by one designer but another would prefer more details.

ACKNOWLEDGEMENTS

This work was funded by the University Technology Partnership for Design, which is a collaboration between Rolls-Royce, BAE SYSTEMS and the Universities of Cambridge, Sheffield and Southampton.

REFERENCES

[1] Fletcher D. and Gu P. Adaptable Design for Design Reuse. In *Second CDEN International Conference on Design Education, Innovation, and Practice*, Canada, 2005.

[2] 80-20 software. 80-20 Retriever Enterprise Edition. http://www.80-20.com/brochures/Personal Email Search Solution.pdf, 2003.

[3] Ye Y. An Active and Adaptive Reuse Repository System. In *the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)*, Vol. 9, Hawaii, 2001.

[4] McCarey F., Cineeide M. O. and Kushmerick N. Knowledge Reuse for Software Reuse, In *International Conference on Software Engineering and Knowledge Engineering*, 2005.

[5] Otto K. and Abecker A. *Corporate Memories for Knowledge Management in Industrial Practice: Prospects and Challenges, In Information Technology for Knowledge Management, 8*, Springer, New York, 1998.

[6] Khadilkar D. and Larry S. An experimental evaluation of design information reuse during conceptual design. *Journal of Engineering Design*, 1996, 7, 4, 331-339.

[7] Burge J. and Brown D. C. *Rationale-based Support for Software Maintenance, Rationale Management in Software Engineering, A. Dutoit, R. McCall, I. Mistrik, and B. Paech*, 2006, Springer

[8] Busby J.S. The Problem with Design Reuse: An Investigation into Outcomes and Antecedents. *Journal of Engineering Design*, 10(2), 1999, 277-296.

[9] Karsenty L. An Empirical Evaluation of Design Rationale Documents. In *SIGCHI conference on Human Factors in Computing Systems*, Canada, 1996, pp.150-156.

[10] Ruben P.D. Reuse as a New Paradigm for Software Development. In *International Workshop on Systematic Reuse*, London, 1996.

[11] Gundry J. and Metes G. Team Knowledge Management: A Computer-Mediated Approach. *A White Paper from Knowledge Ability Ltd*. Manchester NH USA 1996.

[12] Fisher C., Nakakoji K., Ostwald J., Stahl G. and Sumner T. Embedding Critics in Design Environments. *The Knowledge Engineering Review*, 1993, 8(4). 537-561.

[13] Lesh N., Rich C., and Sidner C. L. Using Plan Recognition in Human-Computer Collaboration. In *Seventh International Conference on User Modelling*, Canada, 1999.

[14] Ulrich H. and Schiele F. Towards Task Models for Embedded Information Retrieval. In *SIGCHI conference on Human Factors in Computing Systems*, California, 1992, pp.173-180.

[15] Bracewell R. H. and Wallace K. M. A tool for capturing design rationale. In *14th Int. Conf. on Engineering Design*, Stockholm, 2003, pp.185-186.

[16] Kunz W. and Rittel H. W. J. *Issues as Elements of Information Systems. Working Paper 131*. Center for Planning and Development Research, Berkeley, USA. Elsevier Scientific Publishing Company, 55-169, 1970, Inc. Amsterdam.

[17] Sekine S. and Grishman R. A Corpus-Based Probabilistic Grammar with only two Non-Terminals, In *the Fourth Int. Workshop on Parsing Technologies,* Czech Republic, 2001, pp.216-223.

[18] Miller G.A. Beckwith R.W. Fellbaum C., Gross D. and Miller, K. Introduction to wordnet: An on-line lexical database. *International Journal of Lexicography*, *1993, 3(4)*, 235-312.

[19] Salton G. Advanced Information-Retrieval Models. *In Automatic Text Processing* (Salton, G. Ed.), chapter 10. 1989, Addison-Wesley Publishing Company.

Contact: Dr. Sanghee Kim
University of Cambridge
Engineering Department, Engineering Design Centre
Trumpinton street
Cambridge
U.K.
44-1223-760559
44-1223-332662
shk32@eng.cam.ac.uk

www-edc.eng.cam.ac.uk/people/shk32.html