RECONCILIATION OF KNOWLEDGE – APPLICATIONS IN AUTOMATED DATABASE DESIGN DIGANOSING

Shahrul Azman Mohd. Noah
Faculty of Information Science and Technology,
National University of Malaysia, 43600, Bangi, MALAYSIA
email: samn@ftsm.ukm.my

Michael Williams
School of Computing, University of Glamorgan,
Pontypridd, CF3 1DL, UK.
email: m.d.williams@swansea.ac.uk

ABSTRACT:

The knowledge reconciliation technique to represent real world knowledge has been used by an automated database design tool when performing the task of design synthesis. However, the capacity of such an approach to enhance the diagnostic capabilities of automated database design tools has yet to be explored and evaluated. This paper discusses how the knowledge reconciliation technique has been used during the process of automated database design diagnosing by a prototype tool. Findings suggest that this technique has the potential to improve the diagnostic capabilities of the automated database design tools by facilitating the detection and subsequent resolving of design inconsistencies that would remain undiscovered in situations where system-held real world knowledge was not available.

INTRODUCTION

Automated database design tools employ an artificial intelligence technique in providing assistance to users during the process of database analysis and design. Such assistance can be broadly categorised as design synthesis and design diagnosis (Oxman and Gero, 1987). Design synthesis is where the tools capable of generating design output, whereas, design diagnosis is where the tools detect any inconsistencies and redundancies that may exist and suggest corrections in design. Although there have been a number of automated database design tools emerged from research activities (Lloyd-Williams and Beynon-Davies, 1992; Storey and Goldstein, 1993), human designers consistently outperform such tools at this complex analytical task. Human designers possess what might be called as knowledge of the real world in performing the analysis and design process. For instance, a human designer is capable of diagnosing inconsistencies occurring as a result of using different terms or words for the same concept such as "*client*" and "*customer*", or different forms of the same verb such as "*teach*" and "*taught*", regardless of the application domain. As a result, there have been numerous calls for the representation of real world knowledge within such tools, couple with the ability to reason and make use of this knowledge.

A number of approaches to representing and exploiting real-world knowledge have been proposed, including the *thesaurus* approach (Lloyd-Williams, 1994; 1997), the *dictionary* approach (Kawaguchi et al. 1986) and the *knowledge reconciliation* approach (Storey et al., 1993; 1994). While these approaches have been consistently used in providing assistance to users within the context of design synthesis, their contributions to the tasks of design diagnosis are yet to be explored. This paper presents the implementation and evaluation of the knowledge reconciliation technique to the process of automated database diagnosing using a basic/generic prototype tool as a platform for implementation. The aim is to ascertain as whether such a representation has the potential of being exploited in order to assist users during the tasks of automated database design diagnosing.

OVERVIEW OF AUTOMATED DATABASE DESIGN PROCESSING

In this section, we discuss the activities involved in automated database design. The discussion is based upon a prototype automated database design tool developed to support such activities. The tool under consideration is the Intelligent Object Analyser (IOA) which provides support for the design of the structural (data) aspects of object-oriented databases. It is not the purpose of this paper to discuss IOA in depth, however, a brief outline of the structure and method of operation is required in order to illustrate how the real-world knowledge may be represented and exploited during design processing. Further details of IOA could be found in Shahrul Azman (1999, 2000b). During a design session, IOA follows a two-step procedure.

The first step involves creating an initial representation of the application domain (known as the problem domain model) and the subsequent refinement of this model.

The second step involves the refinement of the problem domain model by detecting and resolving any inconsistencies that may exist, and the transformation of the model into object-oriented form.

The first stage of processing requires a set of declarative statements that describe the application domain to be submitted to IOA. These statements are a variation of the method of interactive schema specification described by Baldiserra et al (1979), being based upon the binary model described by Bracchi et al (1976). Each statement links together two concepts (taking the form *A verb-phrase B*), and falls into one of three classes of construct, corresponding directly to the structural abstractions of association, generalization, and aggregation. The statements are used to construct a problem domain model representing the application domain. Once constructed, IOA attempts to confirm it's *understanding* of the semantic aspects of the problem domain model; that is, whether each structure within the model represents generalization, aggregation or association.

Once constructed, the problem domain model is submitted to a series of refinement procedures in order to detect and resolve any inconsistencies (such as redundancies that may be present within generalization hierarchies) that may exist. These procedures are performed both with and without the requirement of user input (sometimes referred to as *external* and *internal* validation respectively). Once such inconsistencies have been resolved, IOA makes use of the problem domain model in order to generate a conceptual model (in object-oriented form).

There are four types of inconsistencies (errors) that should be detected and resolved by the IOA system. These are as follows.

- *Semantic inconsistencies*. Inconsistencies occurring as a result of missing links (i.e. no associated relationships for a particular concepts) or transitivity that may exists within the generalisation or aggregation hierarchies.
- *Inconsistent concepts*. Inconsistencies occurring as a result of missing properties (i.e. no associated properties for a particular concept).
- *Redundant inherited properties and relationships*. Redundancy occurring within a generalisation hierarchy where a generic class (superclass) and its corresponding specific class(es) (subclass) contain the same properties or participate with the same relationships.
- *Redundant elements*. Inconsistencies occurring as a result of synonyms such as synonymous concepts and relationships which usually lead in turn to redundancies.

The detection and removal of the first three types of inconsistencies is a straightforward process which requires the IOA to inspect each node of the problem domain model and the corresponding links associated with it. Detection of any inconsistency will then brought to the attention of the user for confirmation of its removal. Based upon our testing of the tool during the development process, these types of inconsistencies can be resolved consistently by the tool without the requirement of knowledge of the real world.

However, detection of the fourth type of inconsistency (redundant elements) can be regarded as a complex process. The approach taken by IOA in detecting and resolving this type of inconsistency is based upon the object type and mismatch rules involving the comparison of pairs of association structures of the form "*A association-1 B*" *and* "*X association-2 Y*" within the problem domain model. For instance, if the verb-phrase "*association-1*" is identical to the verb-phrase "*association-2*", and *A* is identical to *X* then the possibility exists that concepts *X* and *Y* are synonymous. The detection of this type of inconsistency, however, is only performed if *A* and *B* and "*association-1*" and "*association-2*" are *exactly* the same. The rule used is as follows.

Rule 1: Let A v_1 B and X v_2 Y be the pair of associations structures, where A, B, X and Y are the concepts and v_1 and v_2 are the verb-phrases. Therefore,

(i) {if A = X and $v_1 = v_2$ then it could be that B is a synonym of Y (B = Y)} or

(ii) {if A = Y and $v_1 = v_2$ then it could be that B is a synonym of X (B = X)} or

(iii) {if B = X and $v_1 = v_2$ then it could be that A is a synonym of Y (A = Y)} or

(iv) {if B = Y and $v_1 = v_2$ then it could be that A is a synonym of X (A = X)}.

Figure 1 illustrates an example of such an inconsistency capable of being detected by IOA. In this example the structures correspond to Rule 1. Therefore the possibility exists that the concepts "*Operation*" and "*Surgery*" are synonymous and considerations of removing one of the redundant concepts can be brought to the attentions of the user.

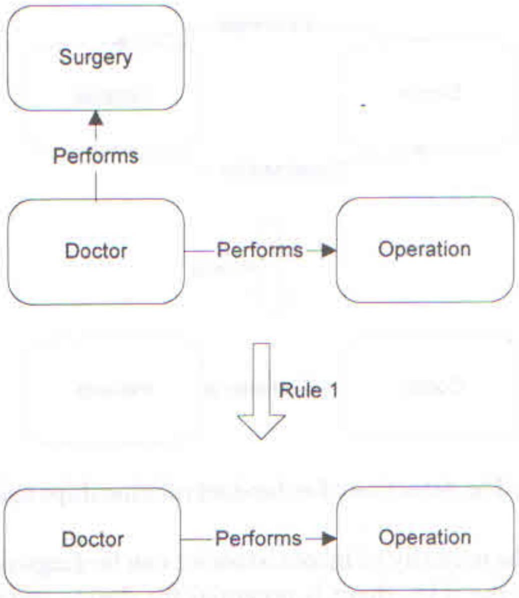

Figure 1: The detection of redundant elements using Rule 1

The object type and mismatch rules are also used to detect and resolve redundant relationships. Such circumstances occur if the concepts *A* and *X* and *B* and *Y* are *exactly* the same. The rule used is as follows.

Rule 2: Let A v_1 B and X v_2 Y be the pair of associations structures, where A, B, X and Y are the concepts and v_1 and v_2 are the verb-phrases. Therefore,
if (A = X and B = Y) or (A = Y and B = X) then it could be that v_1 is a synonym of v_2 ($v_1 = v_2$), or v_1 and v_2 are related by tenses.

Figure 2 illustrates such a situation where pair of association structures can be classified as redundant.

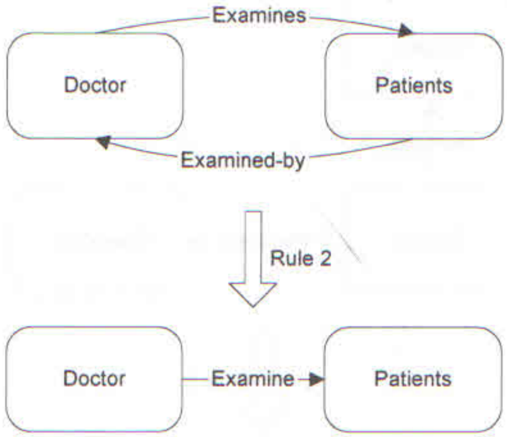

Figure 2: The detection of redundant relationships using Rule 2

Although the majority of inconsistencies can be diagnosed and resolved consistently by the IOA, there is potential for design errors that could be easily identified by a human designer to remain undetected but IOA. For instance, structures such as "*Surgeon Performs Surgery*" and "*Surgeon Conducts Operation*" could not be detected by IOA, since it is unable to detect the similarity implied by "*Performs*" and "*Conducts*". A similar undiagnosed case is represented by the structures "*Doctor Advises Patient*" and "*Patients Consult Physician*". In this case the relationship between the structures "*Advises*" and "*Consult*" and between the concepts "*Doctor*" and "*Physician*" are unknown to the tool.

In the next section of this paper, we will discuss how the knowledge reconciliation technique can be exploited to assist in the detection of such examples of undiagnosed cases.

The Knowledge Reconciliation Technique

The use of the knowledge reconciliation approach in automated database design was initially proposed by Storey et al. (1993, 1994), exhibited in the Common Sense Business Reasoner (CBSR) system which was integrated with an automated database design tool called the View Creation System (VCS) (Storey and Goldstein, 1990a; 1990b). The main purpose of the CBSR system however was to augment the effectiveness of the VCS tool in terms of increasing the capacity of the tool to suggest required missing concepts and relationships.

In order to support this, the approach organises the encapsulated domain knowledge into a collection of domain specific concepts and relationships between these concepts. This knowledge is subsequently reconciled with the user-specified application domain by the system in order to identify any required missing design elements. This reconciliation process employed by the CBSR system includes the reconciliation of concepts and the reconciliation of association relationships (in the form of verb-phrases), which results in the classifications/mappings of concepts and relationships either as *same*, *close* or *different*. For example the concept "*Surgery*" of the user's specified application domain and the concept "*Operation*" of the system's encapsulated domain knowledge could be classified as the *same* because both of the concepts are synonymous.

We will further discuss the knowledge reconciliation technique in the next section.

The implementation of knowledge reconciliation technique

The basic processing of the IOA tool has been previously discussed in this paper. In this section, therefore, we will illustrate how the knowledge reconciliation technique to representing real world knowledge has been implemented in the IOA tool. During implementation, we follow the same approach as those implemented in the CBSR system. However, for those concepts that cannot be classified either as *same*, *close* or *different*, are automatically classified as *unknown* by IOA. The *unknown* classification is just a way of storing information for those concepts that cannot be mapped with the other concepts.

The descriptions of these classifications/mappings are as follows.

- *same(X,α)*. A user specified concept *X* and a real world knowledge concept of α are classified as *same* if the concepts have the same name or they are synonymous;
- *close(X,α)*. A user specified concept *X* and a real world knowledge concept of α are classified as *close* if one concept is a specialisation of the other concept or both are specialisation of a common generic concept;
- *different(X,α)*. A user specified concept *X* and a real world knowledge concept of α are classified as *different* if the user explicitly specified them as different; and
- *unknown(X,α)*. A user specified concept *X* and a real world knowledge concept of α are classified as unknown if the concepts cannot be classified

either as *same*, *close* or *different* (i.e. the IOA unable to identify any concepts that can be mapped with this type of concepts).

Similarly, a pair of verb-phrase relationships are classified as *same* if they are exactly the same, singular/plural of each other or they are synonymous; *close* if they are related by the active/passive form or related by time/tense or *different* if they are explicitly classified different by the user; and unknown if they cannot be classified either as *same*, *close* or *different*.

The above classifications or mappings can be performed by the IOA system automatically or by direct questionings to the user if any elements of uncertainty occurred. From these classifications, the tool is capable of gaining some initial "*understanding*" of the evolving design model as illustrated by Figure 3.

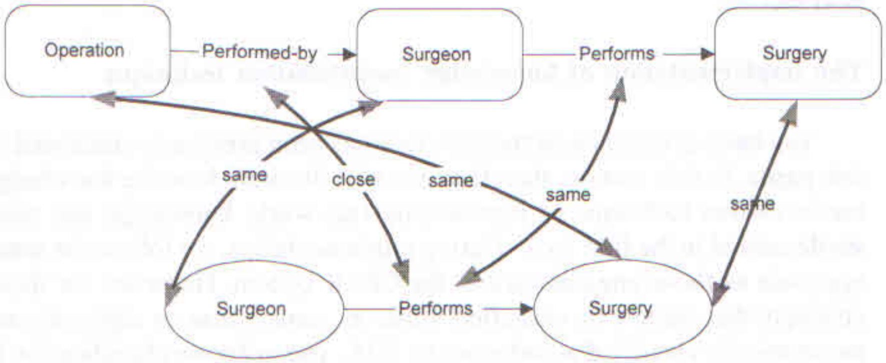

Figure 3: An example of understanding gained by the IOA system after the reconciliation of knowledge process.

The figure illustrates an example of a possible initial understanding gained by the system. It illustrates that the user-concepts "*Surgery*" and "*Operation*" are synonymous because "*Surgery*" has the same mapping with the domain-concept "*Surgery*" and the domain-concept "*Operation*" also has the same mapping with the user-concept "*Surgery*". The figure also illustrates the possible relation between the user-associations "*Performs*" and "*Performed-by*" because of the similar reason as previously stated.

Therefore, apart from the capacity of identifying the potential missing design elements as has been previously issued by the CBSR system, such an understanding also capable of being exploited to enhance the tool's diagnostic capabilities (detecting synonymous concepts for instance as illustrated by the previous example).

The remaining of this section provides discussions of how such capabilities have been exploited.

Exploiting the knowledge reconciliation technique in automated database design diagnosing

As previously mentioned, the detection of redundant elements is performed by the IOA system from the use of Rules 1 and 2. Although the majority of inconsistencies can be diagnosed and resolved consistently by the IOA from these rules, there are three known undiagnosed cases, where relying on Rules 1 and 2 alone fail to detect the inconsistencies. As an illustration, consider a pair of association structures of the form "A v_1 B" and "X v_2 Y", where A, B, X and Y are the concepts and v_1 and v_2 are the verb-phrases. The three undiagnosed cases are as follows.

Case 1: $((A = X) \wedge (v_1 \neq v_2) \wedge (B \neq Y)) \vee ((A \neq X) \wedge (v_1 \neq v_2) \wedge (B = Y)) \vee ((A = Y) \wedge (v_1 \neq v_2) \wedge (B \neq X)) \vee ((A \neq Y) \wedge (v_1 \neq v_2) \wedge (B = X))$. For example "*Academic Teaches Course*" and "*Course Taught-by Lecturer*". This case can be resolved if a tool can at least identify the similarities between the verb phrases or the similarities between the concepts.

Case 2: $((A \neq X) \wedge (v_1 = v_2) \wedge (B \neq Y)) \vee ((A \neq Y) \wedge (v_1 = v_2) \wedge (B \neq X))$. For example "*Lecturer Advises Graduate-student*" and "*Academic Advises Postgraduate-student*". This case can be resolved if a tool can at least identify the similarities between either pair of concepts.

Case 3: $((A \neq X) \wedge (v_1 \neq v_2) \wedge (B \neq Y)) \vee ((A \neq Y) \wedge (v_1 \neq v_2) \wedge (B \neq X))$. For example "*Lecturer Advises Graduate-student*" and "*Academic Consults Postgraduate-student*". This case can be resolved if a tool can at least identify the similarities between either pair of concepts and the similarities between the verb phrases, or if a tool can identify the similarities between both pair of concepts.

However, the initial "understanding" that can be gained from the reconciliation of knowledge previously discussed has the potential of being exploited to resolve such undiagnosed cases. Follows describe how such undiagnosed cases could possibly be resolved from the understanding gained as a result of the knowledge reconciliation process.

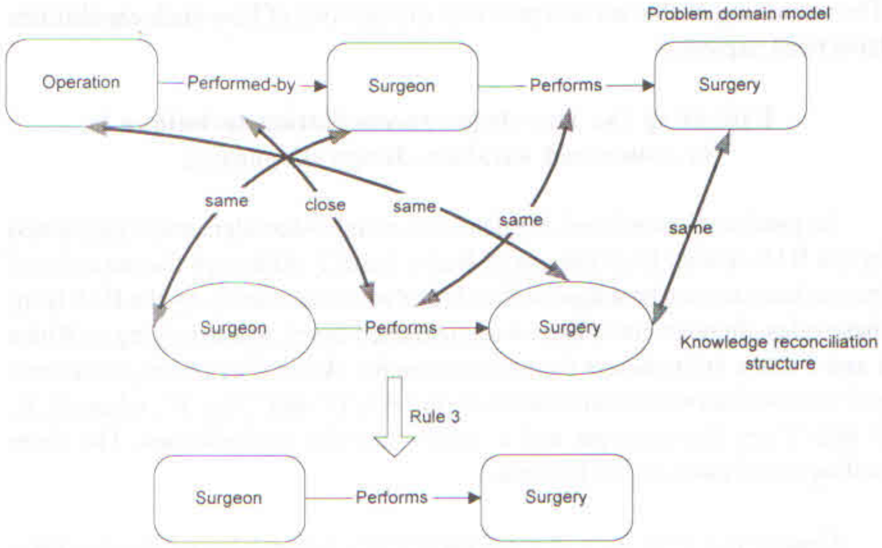

Figure 4: An example of using the understanding gained from the knowledge reconciliation process in the detection and removal of an inconsistency similar to Case 1 (Rule 3)

The initial "understanding" gained from this reconciliation process, allows the possibility to further detect the various types of inconsistencies. Consider an example similar to Case 1, "*Surgeon Performs Surgery*" and "*Operation Performed-by Surgeon*". If appropriate information is available and the required classifications/mappings existed as illustrated by Figure 4, such an inconsistency can be detected and resolved. In the example provided in Figure 4, there has been the same mapping between "*Surgery*" of the real world knowledge concept with "*Surgery*" and "*Operation*" of the user specified concepts. Therefore, there is a possibility that "*Surgery*" and "*Operation*" of the user's specified concepts are synonymous. This conclusion is reached based upon the following rule.

Rule 3: Let A v_1 B and X v_2 Y be the pair of users' specified associations structures (where A, B, X and Y are the concepts and v_1 and v_2 are the verb-phrases) and α γ β as the real world knowledge structure (where α and β are the concepts and γ is the verb-phrase). If *same*(A, α) and *same*(X, α), then it could be that A is a synonym of X (A = X).

A different type of "*understanding*" can also be used to assist in the detection of inconsistencies relating to Case 1. For example consider the structures of "*Physician Examines Patient*" and "*Patient Examined-by Doctor*", and also consider the understanding illustrated by Figure 5.

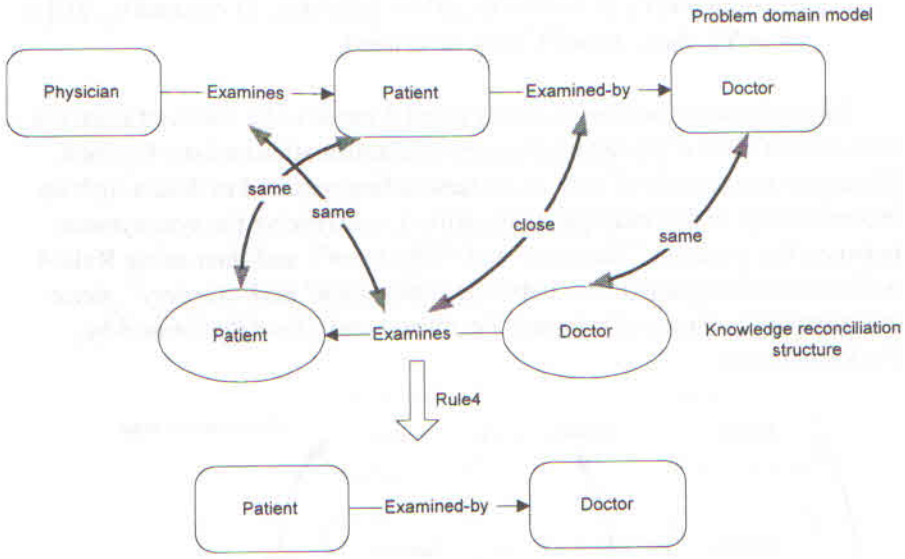

Figure 5: An example of using the understanding gained from the knowledge reconciliation process in the detection and removal of an inconsistency similar to Case 1 (Rule 4)

In this case, the tool knows that the user specified concepts of "*Physician*" and "*Doctor*" may be related since "*Examines*" and "*Examined-by*" are related. This is achieved by the following set of rules.

Rule 4: Let A v_1 B and X v_2 Y be the pair of users' specified associations structures (where A, B, X and Y are the concepts and v_1 and v_2 are the verb-phrases) and $\alpha\ \gamma\ \beta$ as the real world knowledge structure (where α and β are the concepts and γ is the verb-phrase).

(i) If $(A = X) \wedge [(\mathit{same}(v_1, \gamma) \wedge \mathit{same}(v_2, \gamma)) \vee (\mathit{close}(v_1, \gamma) \wedge \mathit{close}(v_2, \gamma)) \vee (\mathit{same}(v_1, \gamma) \wedge \mathit{close}(v_2, \gamma)) \vee (\mathit{close}(v_1, \gamma) \wedge \mathit{same}(v_2, \gamma))] \wedge (B \neq Y)$ then, B and Y may be related, OR

(ii) If $(A = Y) \wedge [(\mathit{same}(v_1, \gamma) \wedge \mathit{same}(v_2, \gamma)) \vee (\mathit{close}(v_1, \gamma) \wedge \mathit{close}(v_2, \gamma)) \vee (\mathit{same}(v_1, \gamma) \wedge \mathit{close}(v_2, \gamma)) \vee (\mathit{close}(v_1, \gamma) \wedge \mathit{same}(v_2, \gamma))] \wedge (B \neq X)$, then, B and X may be related, OR

(iii) If $(B = Y) \wedge [(same(v_1, \gamma) \wedge same(v_2, \gamma)) \vee (close(v_1, \gamma) \wedge close(v_2, \gamma)) \vee (same(v_1, \gamma) \wedge close(v_2, \gamma)) \vee (close(v_1, \gamma) \wedge same(v_2, \gamma))] \wedge (A \neq X)$, then, A and X may be related, OR

(iv) If $(A = Y) \wedge [(same(v_1, \gamma) \wedge same(v_2, \gamma)) \vee (close(v_1, \gamma) \wedge close(v_2, \gamma)) \vee (same(v_1, \gamma) \wedge close(v_2, \gamma)) \vee (close(v_1, \gamma) \wedge same(v_2, \gamma))] \wedge (A \neq Y)$, then, A and Y may be related.

Inconsistencies similar to Cases 2 and 3 can also be resolved from the above set of rules if the required mapping/classifications exists. Figure 6 illustrates an example of such an understanding required to detect such an inconsistency. In this example, using Rule 3, will resolve the synonymous between the concepts "*Surgeon*" and "*Physician*", and then using Rule 4 will resolve the synonymous between "*Operation*" and "*Surgery*", since the mappings of *same*(*Performs*, *Performs*) and *close*(*Performed-by*, *Performs*) exist.

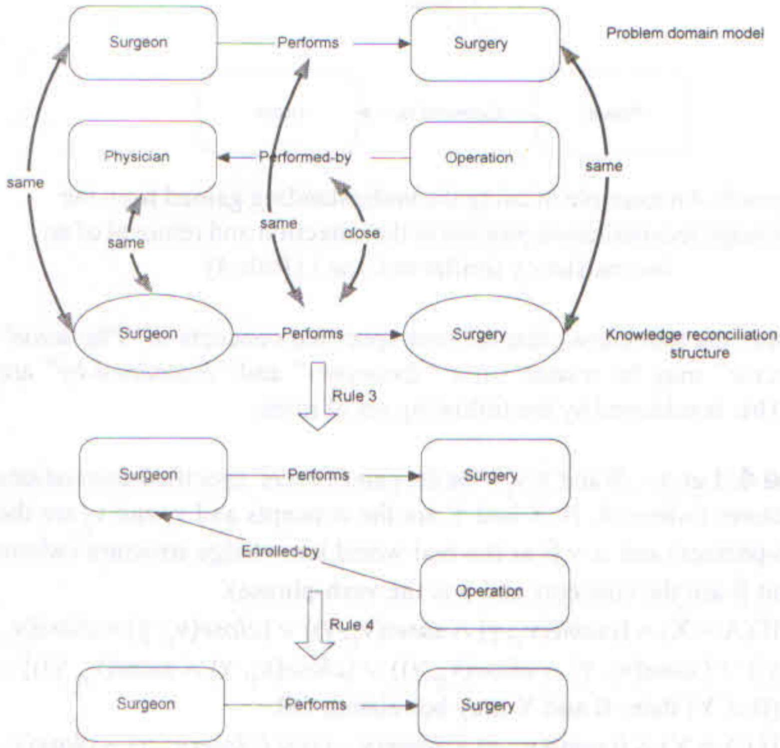

Figure 6: An example of using the understanding gained from the knowledge reconciliation process in the detection and removal of an inconsistency similar to Case 2 and Case 3.

So far, we have only illustrated the significance of the *same* and *close* mapping to the diagnostic activities of automated database design tools. The *unknown* mapping, however, does not make any significant contribution to the diagnostic activities of automated database design tools. As previously mentioned, the mapping is just a way of informing the system's knowledge base for those concepts that cannot be classified either as *same*, *close* or *different* which are crucial for the system's internal inference reasoning. The *different* classification, however, do make some contributions within the aspects of system's intelligence. For example, if structures such as "*Surgeon Performs Operations*" and "*Surgeon Assists Operation*", would be detected by IOA as being potentially redundant (Rule 2), although in reality it would be recognized otherwise by humans. Therefore, a series of interactions with the user will take place. However, if an understanding that the terms "*Performs*" and "*Assists*" are not related (*different*) based from the reconciliation process, the requirements for such a series of interaction will be removed. Further details regarding the implications of the knowledge reconciliation technique to the intelligence aspects of automated database design tool could be found in Noah and Lloyd-Williams (1998) and Shahrul Azman (2000b).

EMPIRICAL TESTING

In order to ascertain as to whether the previously explored knowledge reconciliation approach provides significant contribution to the diagnostic performance of automated database design, a series of testing has been conducted. During the course of conducting this testing, a number of domains were modelled according to the knowledge reconciliation approach. In each case, the real world knowledge structures were developed independently of the example scenarios used during testing. This was a deliberate attempt to minimise any bias that might be introduced by taking the content of the test material into account. In normal circumstances, it would be a logical procedure to use each example application domain encountered to augment the real-world knowledge held, thus, increasing the knowledge of the system in the way a human designer would automatically update his/her knowledge when working within a new domain. However, for the purpose of testing the contribution of real-world knowledge to the diagnostic capabilities of automated database design tools it was decided to develop the knowledge structures independently of the examples encountered.

The case testing method was viewed as being entirely suitable in this experimentation work. The approach taken in this testing, firstly requires the production of a number of synthesised errors. The errors introduced included synonymous concept(s), synonymous or related relationship(s) and a combination of both. Secondly, each of these synthesised errors and combinations of them were systematically embedded into the corresponding design problems to generate the set of test cases.

The set of generated test cases were then executed in IOA with and without the use of real world knowledge (represented as the knowledge reconciliation technique). Thus, for each executed test case, two sets of results were obtained and compared. The design problems used to generate the set of test cases were primarily extracted from existing literature, the advantage being that the accompanying solutions could be used as benchmarks and compared with the IOA-suggested solutions in order to confirm the appropriateness or otherwise of the designs produced (Noah and Williams, 1998). Thus our approach followed that recommended by Liebowitz (1986) and O'Keefe and Preece (1996).

The results presented here emanate from a series of tests performed on university, healthcare and library domain problems found in the general literature. Using the approach previously discussed, a total of 84, 57 and 48 test cases were generated for the university, healthcare and library domains respectively. Using the statistical paired t-test method at the 5% of significance level, the results (number of inconsistencies/errors detected and resolved) from the execution of using the knowledge reconciliation approach were compared with the results from the execution when no real world knowledge was in use. The null hypothesis (H_0) being set up in this test is that there is no significant difference between results from the execution using the knowledge reconciliation technique and results from executions without using the knowledge reconciliation technique, in terms of the number of errors detected and resolved per test. Table 1 illustrates the paired t-test results for all the domains.

Table 1: Paired t-test result – number of errors detected and resolved per test when processing using the knowledge reconciliation technique as compared when processing without using the knowledge reconciliation technique

Domain	**Df**	**t-Value**	**P**
University	83	4.68	0.00
Healthcare	56	2.43	0.01
Library	48	2.54	0.01

It can be seen from Table 1 that consistent results have been obtained and that exploitation of the knowledge reconciliation technique in representing real world knowledge does contribute to enhancing the diagñostic capabilities of automated database design tools. The significant P values and the positive t-Values indicate that the use of the knowledge reconciliation technique presented in this study has significantly increased the number of errors capable of being detected and resolved by the automated database design tool.

CONCLUSIONS AND FUTURE WORKS

This paper has described the method of implementing and exploiting the knowledge reconciliation technique to representing real world knowledge to the process of automated database design diagnosing. Although, existing work on knowledge reconciliation technique has been focused on the task of design synthesis, this paper has revealed that such a representation of real world knowledge also have the potential to play a significant role during the process of design diagnosis. Our findings have demonstrated that the understanding gained from the knowledge reconciliation technique have the capability of resolving inconsistencies caused by using different forms of the same verb as well as different terms or words for the same concepts. However, it is recognised that such capabilities depend greatly upon the accuracy and completeness of the real world knowledge structures presented.

Considering the fact that a consistent and robust conceptual model has a strong influence on the overall likelihood of a successful outcome of a database development project and that the use of system-held real world knowledge by automated database design tool can assist in the development of such models, it can be argued that developers of such tools should give consideration to incorporating elements of real world knowledge into their systems.

Future work includes investigating the possibility of combining the main characteristics represented by the tested approaches to representing real-world knowledge in this study, in order to establish a more reliable and effective real world knowledge model. Another area od future work is to explore other approaches to representing real world knowledge currently implement by intelligent software design tools such as the use of analogy (Maiden and Sutcliffe, 1992), cliché (Reubenstein and Waters, 1991) and design schema (Lubars and Harandi, 1986).

REFERENCES

Baldiserra, C., Ceri, S., Pelagatti, G. & Bracchi, G. 1979. Interactive specification and formal verification of user's views in database design. *Proceedings of the 5th International Conference on Very Large Databases, Rio de Janeiro, Brazil*: 262-272.

Bracchi, G., Paolini, P. and Pelagatti, G. 1976. Binary logical associations in data modeling. In: Nijsen, G. M. (ed.) *Modeling in Data Base Management Systems*. Amsterdam: North-Holland: 125-148.

Kawaguchi, A., Taoka, N., Mizoguchi, R., Yamaguchi, T. and Kokusho, O. 1986. An intelligent interview system for conceptual design of database, in: *ECAI '86: The 7th European Conference on Artificial Intelligence*. London: Conference Services Ltd.: 1-7.

Liebowitz, J. 1986. Useful approach for evaluating expert systems, *Expert Systems*, 3(2): 86-96.

Lloyd-Williams M. 1997. Exploiting domain knowledge during the automated design of object-oriented databases, in: Embley, D W and Goldstein, R C, eds., *Proceedings of the 16th International Conference on Conceptual Modeling*. Berlin: Spinger-Verlag: 16-29.

Lloyd-Williams, M. and Beynon-Davies, P. 1992. Expert system for database design: a comparative review. *Artificial Intelligence Review*, 6: 263-283.

Lloyd-Williams, M. 1994. Knowledge-based CASE tools: improving performance using domain specific knowledge, *Software Engineering Journal*, 9(4): 167-173.

Lubars, M.D. and Harandi, M.T. 1986. Intelligent support for software specifications and design, *IEEE Expert*, 1(4): 33-40.

Maiden, N.A.M. and Sutclifee, A.G. 1992. Analogously-base reusability, *Behaviour and Information Technology*, 11(2): 79-98.

Noah, S.A. and Lloyd-Williams, M. 1998. An evaluation of two approaches to exploiting real-world knowledge by intelligent database design tools, in: Ling, T W, Ram, S and Lee, M L, eds., *Proceedings of the 17th International Conference on Conceptual Modelling*. Berlin: Springer-Verlag: 197-210.

O'Keefe, R.M. and Preece, A.D. 1996. The development, validation and implementation of knowledge-based systems, *European Journal of Operational Research*, 92(3): 458-473.

Oxman, R. and Gero, J.S. 1987. Using an expert system for design diagnosis and synthesis. *Expert Systems: The International Journal of Knowledge Engineering*, 4(1): 4-14.

Reubenstein, H. B. and Waters, R. C. 1991. The requirements apprentice: automated assistance for requirements acquisition, *IEEE Transactions on Software Engineering*, 17(3): 226-240.

Shahrul Azman, M.N. 1999. Knowledge-based CASE tool for object-oriented database design. *Malaysian Science & Technology Congress 1999, Johor Bahru, Johor*, p. 308-315.

Shahrul Azman, M.N. 2000a. Exploring the contributions of knowledge-reconciliation approaches to the diagnostic performance of intelligent database design tools. *ITSim2K, UKM, Bangi*. Bangi-UKM, p.61-69.

Shahrul Azman, M.N. 2000b. Intelligent systems for conceptual modelling of databases. *TENCON 2000*, Kuala Lumpur, 24-27 September 2000. IEEE, p. 504-509.

Storey, V.C., Chiang, R.H.L., Dey, D., Goldstein, R.C., Sundararajan, A. and Sundaresan, S. 1994. Knowledge reconciliation for common sense reasoning, in: De, P and Woo, C, eds., *Proceeding of the 4th Annual Workshop on Information Technologies and Systems*. Vancouver: Univ. British Columbia: 87-96.

Storey, V.C., Goldstein, R.C., Chiang, R.H.L. and Dey, D. 1993.A common-sense reasoning facility based on the entity-relationship model, in: Elmasri, R A, Kouramajian, V and Thalheim, B, eds., *Proceedings of the 12th International Conference on the Entity Relationship Approach*. Berlin: Springer-Verlag: 218-229.

Storey, V.C. and Goldstein R.C. 1993. Knowledge-based approach to database design. *Management Information Systems Quarterly*, 17(1): 25-46.

Storey, V.C. and Goldstein R.C. 1990a. An expert view creation system for database design. *Expert Systems Review*, 2(3): 19-45.

Storey, V.C. and Goldstein, R.C. 1990b. Design and development of an expert database design system. *International Journal of Expert Systems Research and Applications*, 3(1): 31-63.