Using Concept Lattices for Requirements Reconciliation

Debbie Richards

{richards}@ics.mq.edu.au
Department of Computing
Macquarie University, Sydney, Australia

Abstract. We have developed a requirements engineering process model based on viewpoint development. Our goal is to capture and reconcile a set of requirements in the form of use case descriptions so that a more complete and representative set of requirements will provide the basis for system design and development. Our technique converts the natural language sentences in each individual use case viewpoint into a formal context. We then apply Formal Concept Analysis to generate a concept lattice of the requirements. Our process model and tool allow viewpoints, use cases, sentences and phrases to be selected for comparison, tagging and reconciliation. The final product is a shared used case description that has been agreed upon by the project group. In this paper we provide an example developed by final year software engineering students who used the approach as a means to gain experience in the specification of use cases and the sometimes painful task of arriving at a shared view of the requirements before system design and development commenced. Our next step is evaluation with requirements engineers to develop a full-strength industrial product and process.

1 Introduction

We have developed a requirements engineering process model based on viewpoint development. Others (e.g. Darke and Shanks 1997, Easterbrook and Nuseibeh 1996, Finkelstein et al. 1989, Mullery 1979, Kotonya and Sommerville 1998) have also advocated viewpoint development as one possibility to handle some difficulties of requirements definition. What differentiates this work from others is the capture of requirements in natural language (which is far more acceptable and manageable to the average stakeholder than formal specification), automatic generation of a formal representation and the ability to identify and reconcile differences via a visualization of the original text. Formal Concept Analysis plays a key role in the generation and ordering of concepts from use case sentences. We call our approach RECOCASE as we offer a CASE (Computer Aided Software Engineering) tool (as shown in the screen dumps in this paper) to assist with viewpoint RECOnciliation. In this paper we follow an example which uses our requirements engineering process model. At the end we offer some results from initial evaluations, review of related literature and our future directions.

2 The RECOCASE Approach

The RECOCASE methodology includes five iterative phases: requirements capture; requirements translation; concept generation; concept comparison and reconciliation; and evaluation. In this section we follow an example that demonstrates the process and the techniques which we have novelly combined and applied to the problem of requirements reconciliation. The more technical and formal details regarding the use cases, Link Grammar, ExtrAns and RECOCASE-logic are given in (Richards and Böttger 2002). The sample problem we follow is the Automated Video System (AVS) which is being designed to allow customers to self check out videos, search for particular video titles, request new videos to be stocked by each store, and to make reservations for videos currently unavailable. The example is a real one being used by groups of final year software engineering students to develop a software product. We chose this problem as it is a new problem (such systems are not available in Australia) and thus the requirements are not completely understood but also not completely unlike other systems they are familiar with. By following the RECOCASE method-ology students gain some understanding of the difficulties associated with acquiring requirements and managing a number of viewpoints. In the following subsections we take you through each phase. The example uses a reduced set of the actual sentences specified by students as to make the example easier to follow in the limited space available. A whole paper could be written on how to use one diagram to identify and reconcile differences to produce a shared use case.

2.1 Requirements Capture

While a number of approaches to requirements specification have been proposed, see a summary in Düwel (1999), we have chosen to adopt the increasingly popular tech-nique of use case descrip-tion. This first phase starts by a group, led by the group facilitator/leader (GF), brainstorming a set of use cases. Viewpoints for each use case and viewpoint representative are identified. Asynchronously, viewpoint representatives enter use case descriptions in natural language into the RECOCASE tool. Natural language can however be ambiguous and complex in syntax and semantics. A set of writing guidelines and controlled language is provided (Richards, Böttger and Aguilera 2001), but since users rarely follow guidelines, the tool can also detect and notify the viewpoint agent of problematic words, e.g. pronouns, modal verbs, and, or and negations.

2.2 Requirements Translation

The next step is to transform the sentences into a formal representation for later manipulation and comparison. To support comparison via a line diagram using FCA techniques we need to start with a formal context. Each use case forms a new formal context, with the sentences comprising the objects in the context table. Originally we considered a very nave approach which used the words of the sentence as the crosstable attributes. However since sentences are not simply

random strings of words and to allow reconstruction of the sentence by the reader, we are able to use the flat logical forms (FLFs) output by ExtrAns using LinkGrammar as a semantic representation for the sentence which we transpose into word phrases using RECOCASE-logic. For example as shown in figure 1, LinkGrammar[1] provides the following syntactic structure and constituent tree for the sentence 'The system updates the credit card on the member card.'

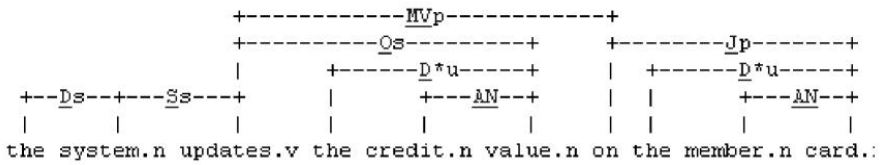

Fig. 1. FLF example

'S' connects the noun 'system' to the finite verb 'update', which is a transitive verb and again connected through 'O' with its object 'value'. 'AN' connects the noun-modifier 'credit' with the noun 'credit'. 'D' connects determiners to nouns. 'system', 'value' and 'card' are in this way connected with the determiner 'the'. 'MV' connects verbs and adjectives to modifying phrases which follow the verb like adverbs, prepositional phrases and subordinating conjunctions. In this example the verb 'update' is connected through 'MV' with the preposition 'on', which introduces a prepositional phrase. To the right the preposition 'on' is connected to its object 'card' through 'J'.

The syntactic structure provided by LinkGrammar is used by ExtrAns to create FLFs as a semantic representation for the core meaning of each sentence.

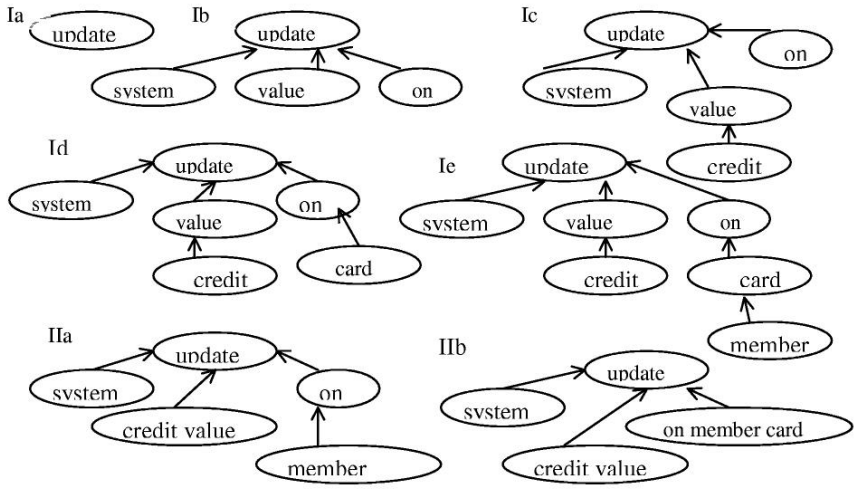

Fig. 2. Steps in the creation (Ia-Ie) and reduction (IIa-IIb) of the sentence "the system updates the credit value on the member card"

[1] Using the parse sentence facility at http://www.link.cs.cmu.edu/link/index.html

The translation of FLFs into crosstable attributes follows an algorithm, known as RECOCASE-logic, which uses graph theory. The algorithm contains two main parts. The first part includes building a graph representing the FLFs of a sentence. The nodes of the graph represent the words of the FLF predicates and the edges represent the relations between them. The relation between words of the predicates is derived from the FLFs. At a coarse level the algorithm is looking for the main event, object or property which is defined by the predicate 'holds'. If the FLF of a sentence does not contain the predicate 'holds', the root 'A' is defined through 'if(A,B)', 'while(A,B)' or 'not(A)' in this sequence. In a second step the graph is reduced to the crosstable attributes. Figure 2 shows the steps involved in creation of the graph for the sample sentence. Table 1 shows the crosstable produced by RECOCASE-logic for the following related sentences from three viewpoints: "The system updates the credit value on the member card", "the system debits the credit of the card" and "the system deducts the price of the rental from the member card".

If all sentences are translated using RECOCASE-logic a crosstable can be created as shown in Table 1. The words or word phrases form the set of attributes where no word or phrase should exist twice. All translated sentences form the set of objects.

	(system)	(update)	(credit value)	(on member card)	(credit)	(debit)	(on card)	(deduct)	(price of rental)
AgentA	×	×	×	×		×			
AgentB	×			×	×	×			
AgentC	×			×			×	×	×

Table 1. Cross table for the sentences "The system updates the credit value on the member card", "the system debits the credit of the card" and "the system deducts the price of the rental from the member card"

2.3 Concept Generation

Figure 3 shows the line diagram for the concept lattice generated from the formal context in Table 1 using standard FCA as described in (Ganter and Wille 1999). As can be seen in the crosstable and the line diagram the only shared attribute is 'system'. The line diagram provides structure and visual clarity of shared concepts. In the next phase we show how the line diagram is used to identify and manage conflicts.

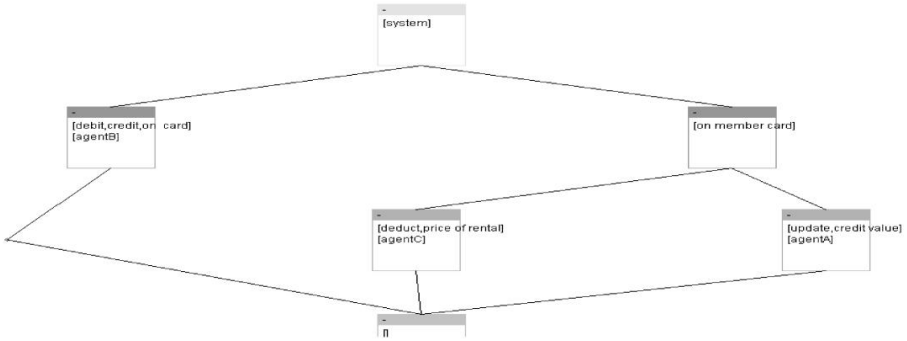

Fig. 3. Representation of the sentences from the context table in Table 1

2.4 Concept Comparison and Reconciliation

The GF creates the diagrams by selecting the sentences he/she wants to have included. Different strategies for combining use case descriptions can be used. If there are few sentences, maybe one or two diagrams constructed by selecting sentences based on system interaction and actors are adequate. However, if there are a larger number of sentences ($\geq$ 25) the GF can choose to create diagrams based on logical units of sentences. The GF will then be able to have a mental overview of the logical flow in the use case. The group members and the GF can use the line diagram to compare viewpoints and detect conflicts. The nodes in the graph represent the concepts, and in our diagrams these are words and phrases from the use case description. The nodes belonging to the same sentence are connected with lines. Sentences containing identical words or synonyms will share nodes.

During the reconciliation process, the group and GF will look at the diagrams together to build a shared use case. Using the lattice we can determine if two concepts are in a state of consensus (same idea, same terms - identified by sharing the same node), correspondence (same idea, different terms - shares some nodes, synonym table needed to map terms), contrast (different idea, same terms - shares some nodes but appears incorrect) or conflict (different idea, different terms - no overlap of pathways). The two key strategies in handling reconciliation is to use a synonym table to map terms and the use of tags to indicate if a concept should be ignored, delayed or circumvented. Figure 4 shows the resulting diagram after some synonyms have been applied to Figure 1. See (Richards 2003) for greater explanation of the tags and our negotiation strategies.

This phase is essentially one involving group discussion under the guidance of the GF. We noted from our evaluation studies that in groups not using the RECOCASE methodology or tool, that participation in this phase was limited to a few interested individuals. It appears that when all participants are required to provide a viewpoint that they tend to engage with the problem and defend their own viewpoint.

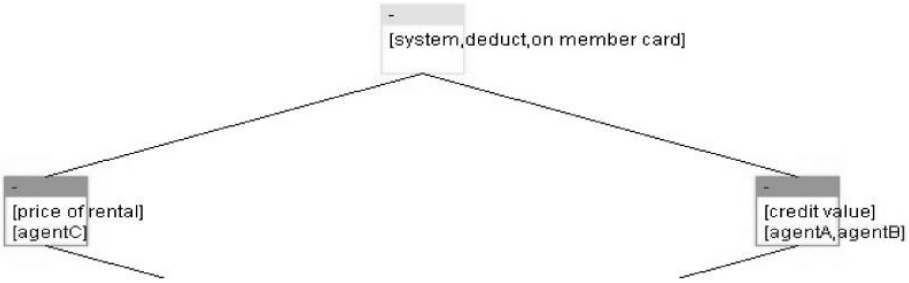

Fig. 4. Representation of the sentences in Figure 3 after a synonym table has been used to map deduct=update=debits, and 'on card' = 'on member card'.From this figure we can see we still need to map 'price of rental' = 'credit value' to bring all viewpoints into consensus

2.5 Evaluation

We use graph theory on the lattices to evaluate if the different viewpoints have become more similar, or if a new round of negotiation is necessary. In the RECOCASE tool we can select viewpoints for comparison and see the percentage of similarity between descriptions. We can also see the percentage of similarity between each sentence in two different viewpoints. This information can be used to determine which viewpoints to try to reconcile first (e.g. reconcile most similar or most different first).

3 Discussion and Conclusion

We have conducted a number of different evaluations. Our first studies concentrated on the learnability and comprehensibility of the line diagram for reasoning about requirements. We found that after a 5 minute introduction, 58% of students were able to correctly interpret (no errors) and 80% more likely to give the correct answer to questions involving comparison of viewpoints using the diagram as opposed to the original text. A particularly significant difference was the time taken to answer questions using the diagram rather than text was up to (a number of tests were done) 9.9 times faster. Our recent studies evaluating the group process and tool revealed that each final and shared use case description was longer than the individual use case descriptions. All participants were satisfied with the process and the outcome and were able to create a shared use case description using the technique. The most striking difference was found in the attitudes and cohesion of the groups that used the process compared to the groups that did not.

The application of FCA for software engineering activities has been pursued by a number of researchers. A useful survey is provided in (Tilley et al. 2003). Whilst the application of FCA to the requirements phase has not been the primary area of activity targeted some work does exist. Andelfinger's (1997) thesis, reported in Tilley et al. 2003, concerned a question-answering and discussion tool using FCA which could be applied to the area of requirements capture. More

directly relevant is the work of Düwel and Hesse (2000) who use FCA to identify potential classes and attributes from use case descriptions. Their work is different to ours in a number of ways. Firstly, in the formal context they build, the use cases are the columns (attributes) and the rows (objects) are the potential classes or attributes. In our approach we develop one or more formal contexts for each use case, with objects being sentences and attributes being words and phrases, and the formal context usually contains at least two viewpoints of the complete or partial use case. Thus our focus is much more fine-grained and not restricted to certain parts of the sentence. Düwel and Hesse have also developed a tool to assist with entry of use case descriptions and development of concept lattices. As with our tool, they shield the user from the mathematical complexities of FCA. Also, as we do, they acknowledge that the decision process, which in their case concerns the choice of classes to use and how they relate to one another, "requires much detailed work which cannot completely be automated, including negotiations and a thorough discourse between domain experts and software developers". However, the approach to creation of the formal context differs significantly from ours. Our approach takes in natural language sentences and automatically generates a formal context using natural language techniques. In their approach, the user is required to select words from a list or the description which become the set of objects. Given that our approach identifies nouns, objects and events (all possible candidates for classes), we envisage that our natural language techniques could be beneficially combined with the Düwel and Hesse approach. Indeed we believe that our ability to produce FLFs from the natural language descriptions offers a entry point and input solution for many other formal specification techniques.

As a result of the initial success of the approach with students, which has led to revision of the tool and fine tuning of the process substeps, we are seeking industrial collaboration to develop the approach further and incorporate it as an integral step of object oriented design using UML. As Düwel and Hesse (2000) stated "we find it quite an interesting observation that just a "non-OO" technique like use case analysis has become so popular as a starting point to OO analysis", we also find it interesting that use case descriptions are textual and the rest of the UML process is visual. By offering an approach which visualizes the textual descriptions we provide a completely visual technique to system specification and design. Further, by capturing the requirements from multiple viewpoints we get a more complete and consistent set of requirements and a greater sense of ownership by the project group, thus providing a better basis for system design and increasing the likelihood of eventual system acceptance.

Acknowledgment

Many thanks to those who have contributed to this project: Oscar Aguilera, Kathrin Böttger, Anne Britt Fure, Rolf Schwitter and Diego Molla-Alloid. This project is funded by the Australian Research Council.

References

1. Darke, P., Shanks, G.: Managing user viewpoints in requirement definition. 8^{th} Australasian Conference on Information Systems. 1995.
2. Düwel, S. (1999) Enhancing System Analysis by means of Formal Concept Analysis In Conference on Advanced Information Systems Engineering 6th Doctoral Consortium, Heidelberg, Germany.
3. Düwel, S and Hesse, W. (2000) Bridging the gap between Use Case Analysis and Class Structure Design by Formal Concept Analysis In J. Ebert and U. Frank (eds.) Modelle und Modellierunssprachen in Informatik and Wirtschaftsinformatik. Proceedings "Modellierung 2000", Koblenz, 2000, Foelbach-Verlag, 27-40.
4. Easterbrook, S. and Nuseibeh, B. (1996) Using Viewpoints for Inconsistency Management BCSEEE Software Engineering Journal, January 1996, 31-43.
5. Finkelstein, A.C.W., Goedicke, M., Kramer, J. and Niskier, C. (1989) Viewpoint Oriented Software Development: Methods and Viewpoints in Requirements Engineering In Proc. of the 2nd Meteor Workshop on Methods for Formal Specification, Springer Verlag, LNCS.
6. Ganter, B. and Wille, R. (1999) Formal Concept Analysis: Mathematical Foundations, Springer, Berlin.
7. Kotonya, G. and Sommerville, I. (1998) Requirements Engineering: Process and Techniques Chichester, UK, Wiley and Sons.
8. Mullery, G. P. (1979) CORE - a method for controlled requirements expression. In Proceedings of the 4th Int. Conf. on Software Engineering (ICSE-4), IEEE Computer Society Press, 126-135.
9. Richards, D (2003) Merging Individual Conceptual Models of Requirements, Spec. issue on Model-Based Requirements Engineering for the International Journal of Requirements Engineering.
10. Richards, D., Böttger, K and Aguilera, O. (2002) A Controlled Language to Assist Conversion of Use Case Descriptions, 15th Australian Joint Conference on Artificial Intelligence, Canberra, Australia, December 2nd - 6th, 2002.
11. Richards, D. and Böttger, K. (2002) Representing Requirements in Natural Language as Concept Lattices, In Proc. 22nd Annual Int. Conference of the British Computer Society's Specialist Group on Artificial Intelligence (SGES), (ES2002), Cambridge, UK, December 10-12, 2002
12. Tilley, T., R. Cole, P. Becker and P. Eklund, A Survey of Formal Concept Analysis Support for software engineering activities", in *Proceedings of the First International Conference on Formal Concept Analysis - ICFCA'03*, (eds) G. Stumme and G. Ganter, Springer Verlag, February 2003 (*to appear*).
13. Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In Reidel, D. Ordered Sets, Dordrecht, 1982, pp. 445–470.
14. Wille, R.: Concept lattices and conceptual knowledge. Computers and Mathematics with Applications **23** (1992) 493–522.