RECOCASE-tool: A CASE tool for RECOnciling Requirements Viewpoints

Kathrin Boettger, Debbie Richards and Anne Fure

Department of Computing
Division of Information and Communications Sciences
Macquarie University
Sydney, Australia
Email: richards@ics.mq.edu.au

Abstract
Viewpoint development has been proposed as one way to develop a set of requirements that are representative of the many stakeholders who may be involved with a software project. This paper describes a tool developed to support our viewpoint development approach. A key feature of this work compared to most viewpoint research is that we do not assume we have requirements in computer readable form. Neither do we expect formalisation of requirements to be performed by logicians or others trained in formal specificiation methods. We commence with brainstorming use cases and identifying viewpoints. Each viewpoint agent enters a description for each use case into the RECOCASE-tool. LinkGrammar is applied to translate the sentences of the use case descriptions into flat logical forms (FLFs). FLFs are used to create crosstables. Formal Concept Analysis is used to develop a graphical representation of the viewpoints. We then apply the various resolution strategies and operators that are part of our framework to develop a shared conceptual model of the requirements.

Keywords
CASE Tool, viewpoint development, formal concept analysis, use cases

1. Tool Overview

Viewpoint development has been proposed (e.g. Darke and Shanks 1997, Easterbrook and Nuseibeh 1996, Finkelstein et al 1989 and Mullery 1979) as one way to develop a set of requirements that are representative of the many stakeholders who may be involved with a software project. This paper describes a tool developed to support our viewpoint development approach. A key feature of this work compared to the other viewpoint research cited is that we do not assume we have requirements in computer readable form. Neither do we expect formalisation of requirements to be performed by logicians or others trained in formal specificiation methods. As advocated by many object-oriented analysis and design textbooks, we commence with brainstorming the main chunks of functionality from the user's point of view, that is, we identify use cases (Jacobson 1992). Then using our tool we create a project and add the use case names. Individual stakeholders representing a viewpoint enter their requirements by specifying descriptions for each use case and/or scenario. These descriptions may be in natural language but better results are achieved with our tool when a controlled language is used. We have developed guidelines and tool support to assist the user in complying with the controlled language (Boettger *et al.* 2001). LinkGrammar is used by an answer extraction system (ExtrAns) to translate the sentences of the use case description into flat logical forms (FLFs). FLFs are used to create crosstables. Formal Concept Analysis (FCA) (Wille 1982, 1992) is used to develop a graphical representation of the viewpoints. We then apply the various resolution strategies and operators that are part of our framework to develop a shared conceptual model of the requirements.

We call our approach RECOCASE as we offer a CASE (computer aided software engineering) tool to assist with viewpoint RECOnciliation. Our framework is not discussed in this paper as it was introduced in (Richards and Menzies 1998, Richards and Zowghi 1999). This paper focuses on the prototype RECOCASE-tool. The tool's functionality is represented in the use case diagram in Figure 1. The tool supports the composition of viewpoints and the translation of the sentences of these viewpoints into crosstables. Crosstables are the input for FCA to create concept lattices (use case: 'create input for FCA'). Before someone can compose viewpoints a viewpoint model has to be created (use case: 'create viewpoint model'). It is possible to compose use case viewpoints (use case: 'create use case viewpoint') and scenario viewpoints (use case: 'create scenario viewpoint'). Before one can compose or analyse viewpoints one has to login into the system (use case: 'verify user') to control which actions someone can perform using the tool. We have identified two actors. The analyst role could be a group facilitator or team leader and does not necessarily need to be a requirements analyst. However, access to all viewpoints, set up of the project and manipulation of the shared viewpoint has been restricted to a role that we have identified by the name 'analyst'. A stakeholder is a viewpoint agent who has ownership and responsibility for their own viewpoint use case descriptions and/or scenarios.

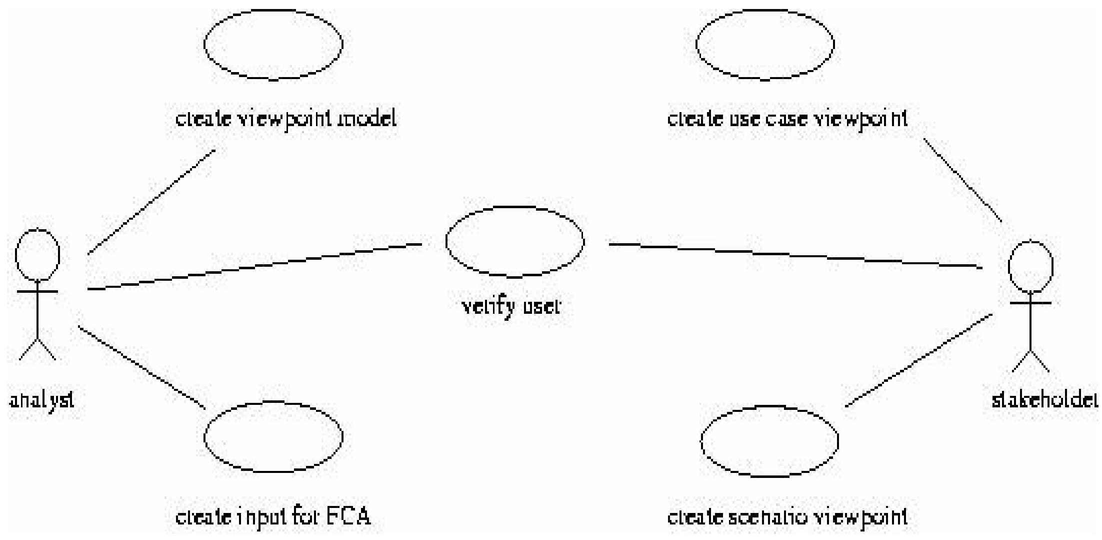


Figure 1: Use case model of RECOCASE-tool

RECOCASE-tool has been written in Java. To be able to use LinkGrammar, which is written in C, an interface has been written using Java Native Methods and linked to the program as a library. ExtrAns is written in Sicstus-Prolog. The Jasper-package of Sicstus-Prolog, an interface for the use of Sicstus-Prolog in Java programs, is used to access a file written in Prolog that calls Extrans to produce FLFs. An interface has been written using Java Native Methods to use a program that has been written in C which processes the FLFs to RECOCASE logic forms. The program is linked as a library. All libraries were created for a Unix platform, because ExtrAns and ExtrAns connected tools are executable only on a unix platform. The part of the RECOCASE-tool, that is used to compose viewpoints, also runs on a Windows platform.

In addition a database model has been developed and implemented in MS Access 2000 to store the viewpoint data. RECOCASE-tool is connected with the database through the network using the aveConnect JDBC driver.

Figure 2 shows the business model of RECOCASE-tool. For each viewpoint model a project ('CProject') must be created. Each shared use case viewpoint ('CUsecase') is attached to one project thus to one viewpoint model. Each shared scenario viewpoint ('CScenario') is attached to one shared use case viewpoint. All actors of the viewpoint model have to be attached to the corresponding project.

All individuals or groups of people who are using the system are represented by users ('CUser'). Each user can create more than one viewpoint ('CViewpoint'). There exist use case viewpoints ('CViewpointUc') and scenario viewpoints ('CViewpointSc'). Each use case viewpoint is attached to one shared use case viewpoint and each scenario viewpoint is attached to one shared scenario viewpoint. An actor of a shared use case viewpoint has to be an actor of the project.

The next section describes how the tool can be used for the individual phases of RECOCASE.

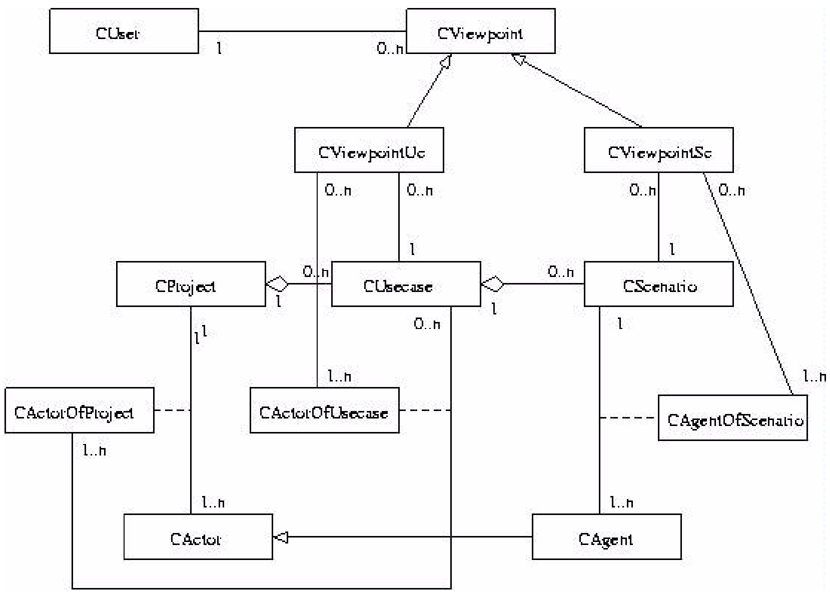


Figure 2: Business Model of RECOCASE-tool

2. The Viewpoint Development Process

The RECOCASE development process includes six iterative steps. For the purpose of this paper we have adapted our process to the Viewpoint Development Process proposed by Darke and Shanks (1997) which includes the phases of viewpoint identification, viewpoint representation, intra-viewpoint analysis, inter-viewpoint analysis and viewpoint integration. We consider each of these phases next.

2.1 Viewpoint Identification

Viewpoint identification starts with the creation of a viewpoint model which is an extension of Jacobson’s use case model (1992). All methods for the identification of Jacobson’s use case model can be applied to the identification of a viewpoint model. Jacobson (1992) suggests an actor-based method for identifying use cases, which means to define first what is outside the system (actors) and to investigate then the system functionality (use cases). The last means to identify what users are able to do as an occurrence of an actor. Another possibility is to identify first the external events that a system must respond to and then to relate the events to actors and use cases. Our preferred technique is to identify a representative for each actor. The representative or ‘viewpoint agent’ is responsible for describing that viewpoint. In our approach the use case model represents the shared use case viewpoint model, which covers the shared scenarios viewpoints. When using the tool, a project is first created by specifying a name, a general description of the functionality and assigning actors to the project. If the project is open the analyst can create the shared use case viewpoints. To create shared scenario viewpoints the shared use case viewpoint to which they shall be assigned must be open.

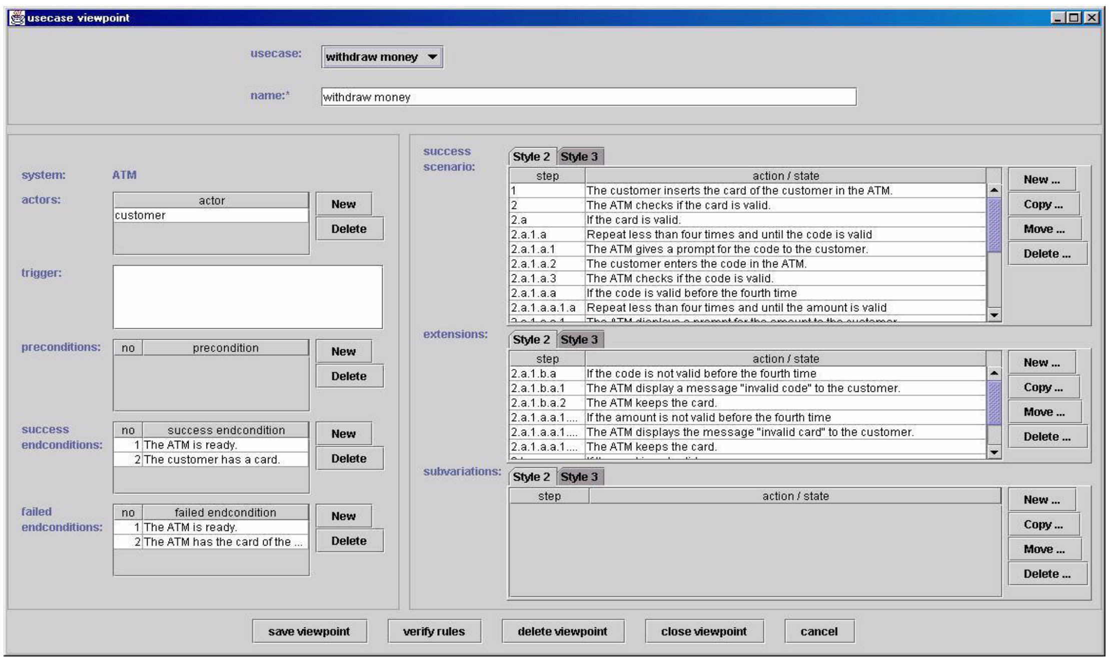
2.2 Viewpoint Representation

RECOCASE captures viewpoints of functional requirements in the form of use cases and scenarios. The use case viewpoints are divided into several parts: actors, trigger, pre- and postconditions and the flow of actions. The postconditions are again divided into success postconditions and failed postconditions. The flow of actions is again divided into a main success scenario, an extension part of the main success scenario and a variation part of the main success scenario. Scenario viewpoints consist of agents and the flow of actions.

Our tool provides three alternative ways of entering and structuring use case descriptions. One possible way to describe the flow of actions is to use unstructured text (-style 1-). Cox (2001) suggests to write a use case as a list of discrete actions in the form <action#> <action description> and to use a separate line for each action (-style 2-). Wirfs-Brock (1993) proposed a structured form which is divided into a user-action-model and a system-response-model to describe the interaction between a user and a system through a graphical user interface (-style 3-). See Table 1 for an example. The user-action model represents what the user does and the system-response model shows the system’s responses to the user actions. This model is taken up by Constantine and Lockwood (1999). For the approach described in this paper the model of user-system interaction by Wirfs-Brock is extended to a model of actors-system interaction to be able to describe the interaction between the system and more than one actor. The graphical user interfaces for use case and scenario composition are given in figures 3 and 4, respectively. As shown in these figures, scenarios may be specified in any of the 3 styles. Use case descriptions may be written in style 2 or 3 since the free format of style 1 is too unstructured for our tool to enforce the guidelines and controlled language.

User Action	System Response
1. Insert card	2. Read magnetic stripe
	3. Request PIN
4. Enter PIN	5. Verify PIN
	6. Display transaction option menu
7. Press key	8. Display account menu
9. Press key	10. Prompt for amount
11.Enter amount	12. Display amount
13. Press key	14. Return card
15. Take card	16. Dispense cash
17. Take cash	

Table 1: Model of user-system interaction modified from Wirfs-Brock (1993)



usecase: withdraw money

name:* withdraw money

system: ATM

actors:

actor
customer

trigger:

preconditions:

no	precondition

success endconditions:

no	success endcondition
1	The ATM is ready.
2	The customer has a card.

failed endconditions:

no	failed endcondition
1	The ATM is ready.
2	The ATM has the card of the ...

success scenario:

step	action / state
1	The customer inserts the card of the customer in the ATM.
2	The ATM checks if the card is valid.
2.a	If the card is valid.
2.a.1.a	Repeat less than four times and until the code is valid
2.a.1.a.1	The ATM gives a prompt for the code to the customer.
2.a.1.a.2	The customer enters the code in the ATM.
2.a.1.a.3	The ATM checks if the code is valid.
2.a.1.a.a	If the code is valid before the fourth time
2.a.1.a.a.1.a	Repeat less than four times and until the amount is valid

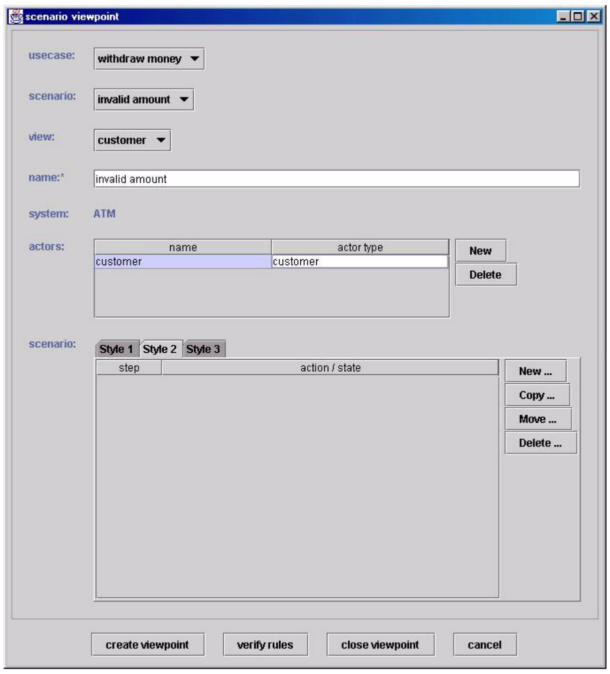
extensions:

step	action / state
2.a.1.b.a	If the code is not valid before the fourth time
2.a.1.b.a.1	The ATM display a message "invalid code" to the customer.
2.a.1.b.a.2	The ATM keeps the card.
2.a.1.a.a.1....	If the amount is not valid before the fourth time
2.a.1.a.a.1....	The ATM displays the message "invalid card" to the customer.
2.a.1.a.a.1....	The ATM keeps the card.

subvariations:

step	action / state

Figure 3: Graphical User Interface for Composition of a Use Case Viewpoint


scenario viewpoint

usecase: withdraw money

scenario: invalid amount

view: customer

name:* invalid amount

system: ATM

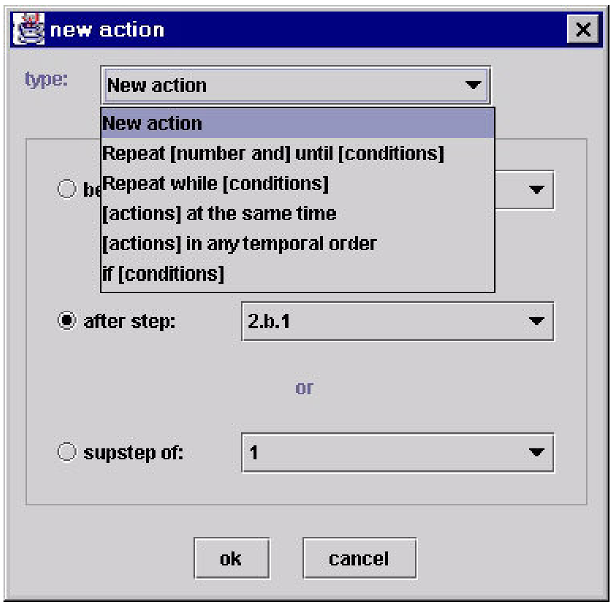
actors:

name	actor type
customer	customer

scenario: Style 1 Style 2 Style 3

step	action / state

Figure 4: GUI for Composition of Scenario Viewpoint


new action

type: New action

- New action
- Repeat [number and] until [conditions]
- Repeat while [conditions]
- [actions] at the same time
- [actions] in any temporal order
- if [conditions]

after step: 2.b.1

or

supstep of: 1

Figure 5: Definition of a new Scenario Step

Before viewpoint agents can compose their viewpoints they have to log into the system and open the corresponding project. Then they can choose to create or open a use case viewpoint or a scenario viewpoint. A viewpoint agent can save and delete any viewpoints which he or she created. Only users with analyst access may modify the viewpoints of other agents. The tool assists the viewpoint agent in the creation of steps for the flow of action. The agent can choose from a list which kind of step he or she wants to describe (see Figure 5). The tool creates the number of steps automatically. The viewpoint agent can also delete, copy or move existing steps. The tool will correct the flow of actions automatically.

2.3 Intra-Viewpoint Analysis

After a viewpoint is described it should be checked to see if the viewpoint agent followed the guidelines (Boettger *et al.* 2001). This is important as we need to translate the natural/controlled language sentences into tabular format. In RECOCASE-tool we provide manual and automatic checking. The user can press the 'verify rules' command button to have their sentences checked before they request conversion of the sentences into a crosstable. To conform with our use case description guidelines, the tool looks for unknown words, modal verbs, personal and possessive pronouns and replaces them or asks the user to provide an alternative. Alternatively the user can select 'create viewpoint' without first verifying the sentences. Step by step verification by the viewpoint agent is preferred as it avoids errors that may be harder for the user to identify and correct later and the process also assists the viewpoint agent in learning the controlled language. We are currently working on adding further verification features to the tool at the word and sentence level. These features directly relate to the guidelines and rules of the controlled language. By enforcing the controlled language the concept lattice can be improved as a graphical representation of the controlled language sentence.

Tool support for the controlled language may be provided as follows. The guideline code is given in square brackets and a description given in a footnote. The guidelines are given in full in (Boettger 2001).

Identification of 'synonyms', 'hyponyms' and 'hypernyms' can increase the similarity of concepts and improve comparison of viewpoints and readability of a concept lattice [W1[1]]. Only small improvements might be achieved by using a dictionary valid for all domains like 'Wordnet' since there are no perfect synonyms because no two words ever have exactly the same meaning (Fromkin 1996, p.131). However dictionaries exist for some domains which can provide significant improvements. Concept lattices can themselves to used to find similar terms due to the extensional and intensional definition of formal concepts in FCA. More is said about this later.

LinkGrammar offers several possibilities to parse a sentence and to use the LinkGrammar output for processing viewpoints. The ways in which LinkGrammar could be used to assist the viewpoint agent in writing viewpoints in the controlled language are now described.

The output of LinkGrammar's word segmentizer can be used to check if phrases like personal pronouns (e.g. he, she, it), possessive pronouns (his, her, its, hers) modal auxiliaries (e.g. can, shall), semicolons, dashes and colons are used according to [W3-W5[2]].

Slang, abbreviations and symbols should not be used [W6-W7[3]]. For the word segmentizer of LinkGrammar unknown words are marked as keywords. The words may be unknown because the spelling is incorrect or the word is not held in the dictionary of LinkGrammar. In the last case there exists the possibility to extend the dictionary or to choose another word to ensure that the outcome of further steps of language processing is correct.

The tool could give advice how the viewpoint agent should compose a sentence so that it is formatted according to [S2[4]] which is:

sentence: <subject> <predicate>,
predicate: <predicator> <complement(s)>

[1] [W1] The viewpoint author should use words in a consistent way and should therefore avoid the use of synonyms, hyponyms and hypernyms. Why: To reduce the number of nodes of the concept lattice.

[2] [W3] The viewpoint author should not use personal pronouns (e.g. he, she, it) and possessive pronouns (his, her, its, hers). Why: Pronouns are notoriously difficult to resolve since the search space for the correct noun might be very deep.

[W4] The viewpoint author should not use modal verbs (e.g. might, could, should). Why: Modal verbs express doubt.

[W5] The viewpoint author should not use semicolons, dashes and colons. Why: RECOCASE is using tools (see section 2.4 and 2.5) for natural language processing. These tools are still not able to process sentences containing semicolons, dashes and colons.

[3] [W6] The viewpoint author should not write slang. Why: They are often not contained in dictionaries.

[W7] The viewpoint author should not use symbols and abbreviations instead of words and phrases. Why: They are often not contained in dictionaries.

[4] [S2] An action should be described in one grammatical form - a simple action in which one actor either accomplishes a task or passes information to another actor. A simple sentence has the general structure:
sentence: <subject> <predicate>
predicate: <predicator> <complement(s)>
A sentence should contain one subject, one verb and depending on the verb zero (intransitive verbs), one (transitive verbs) or two (ditransitive verbs) complements.

LinkGrammar can separate the sentence into noun and verb phrases. This can be used to transpose composed sentences into simple sentences according to [S2] in the following way. The following sentence structures can be easily converted into a phrase structure tree.

- Sentence (S) with two noun phrases (NP) and once verb phrase (VP) connected by 'and': for example LinkGrammar provides for the sentence 'The customer inserts the card and the code.' the following structure:

```
(S (NP The customer)
 (VP inserts
      (NP  (NP the card)
            and
            (NP the code)))
```

The tool could suggest to split up the sentence into two sentences 'The customer inserts the card.' and 'The customer inserts the code.' or to add the phrase 'at the same time' if the author wants to express that the actions happen simultaneously.

- Sentence with two verb phrases connected by 'and': For example LinkGrammar provides for the sentence 'The ATM recognizes the card and asks for the code.' the following structure using a prepositional phrase (PP):

```
(S (NP The ATM)
   (VP (VP recognizes
         (NP the card))
   and
   (VP asks
      (PP for
         (NP the code))))
 .)
```

The tool could suggest to split up the sentence into the two sentences 'The ATM recognizes the card.' and 'The ATM asks for the code.' or to add the phrase 'at the same time' if the author wants to express that the actions happen simultaneously.

- Sentence composed of simple sentences: For example LinkGrammar provides for the sentence 'After the customer inserted the code the ATM checks the code.' the following structure:

```
(S″ (S′ After
     (S (NP the customer)
      (VP inserted
      (NP the code))))
  ,
     (S (NP the ATM)
      (VP checks
      (NP the code)))
  .)
```

The tool could suggest to split up the sentence into the two sentences 'The customer inserts the code.' and 'The ATM checks the code.' or to add the phrase 'at the same time' if the author wants to express that the actions happen simultaneously.

The syntactic structure provided by LinkGrammar which is used for further natural language processing (e.g. by ExtrAns) could be used to explain the sentence structures to the viewpoint agents. Two interpretations exists of how the prepositional phrase 'with a PIN number' is associated with the rest of the sentence for the sentence 'The ATM customer inserts the ATM card with a PIN number.' Is the prepositional phrase connected with the verb 'insert' and does it mean 'together with the PIN number' or could it mean that the ATM card has a PIN number? Prepositional phrases should be used to modify a verb [S11[5]]. The tool can request clarification where this has not been done.

The linkage type 'S' of the syntactic structure connects subject nouns to the finite verbs. By using this it is possible to check guideline [S3[6]], if the subject is the system or an actor. Sentences should be written in the active, not the passive voice [S6[7]]. If the FLF of a sentence provides the predicate 'hearer' then the sentence might be an imperative sentence. The predicate 'anonym_object' refers to a passive sentence, which does not define an agent. In other cases the viewpoint agent could be asked to change the sentences. For example the FLF of the sentence 'The card is checked' contains the predicate 'anonym_object'. The viewpoint agent should change the sentence to 'The ATM checks the card.'. Thus, Extrans can be used to check if viewpoints contain imperative sentences or passive sentences which do not define an agent.

At the current stage the tool assists in finding words which are unknown for ExtrAns or which are treated as keywords. The tool also provides an output referring to the structure of each sentence (noun phrases, verb phrases, phrasel sentences).

2.4 Inter-Viewpoint Analysis and Viewpoint Integration

The previous section was concerned with the internal consistency of a viewpoint and is a necessary prerequisite for inter-viewpoint analysis and integration into a shared viewpoint which is the ultimate goal of this work. The concept lattices could also be used as a creativity tool to assist in a brainstorming session and as a communication tool to discuss concepts and the relationship between them.

To identify similarities and differences between formal concepts we apply the four quadrant model of comparison developed by Shaw and Gaines (1988) which defines the four states of Consensus, Correspondence, Conflict and Contrast. As seen in Figure 6, the states are distinguished according to whether the same concept/idea and/or same terminology are being used. In our case a concept could be an event, use case step, sentence, etc. We distinguish between a concept and a formal concept as a formal concept in FCA is a pair comprised of a set of objects and the set of attributes shared by those objects. In the FCA sense, we treat each sentence as an object and the parts of the sentence as the attributes which make up that object.

[5] [S11] The viewpoint author should use prepositional phrases to modify a verb and avoid the use or relative clauses, which should be used to modify a noun. Why: A prepositional phrase can sometimes be associated with the verb or with a noun. The human can often use information from the context to decide which interpretation is meant, but it is difficult to use the computer to do this. Another reason is to standardize the concept lattice.

[6] [S3] The subject should be the system or an actor. Why: To make the structure and content of concepts consistent.

[7] [S6] The viewpoint author should use active voice instead of passive voice. Why: It is important to know who triggers the action.

State	Concept	Terminology
Consensus	Same	Same
Correspondence	Same	Different
Conflict	Different	Same
Contrast	Different	Different

Figure 6:Four Quadrant Model of Comparison (Shaw and Gaines 1988)

If all viewpoint agents described their viewpoints at the same level of detail a concept would always be described by a sentence. This would be the case because RECOCASE assumes that each action or state of the flow of action is described by a sentence represented as a formal concept in FCA and because each action can be considered as separated from other actions. But for example one viewpoint agent describes the notion of 'checking the ATM card' by 'The ATM checks the ATM card.' and another viewpoint agent describes this process using the three sentences 'The ATM checks if the date is expired.', 'The ATM checks if the ATM card was stolen.' and 'The ATM checks if the ATM card was lost.'. Therefore sets of concepts have to be compared.

The first step of this phase involves domain modeling by analysing the terminology. We want to determine if the viewpoint agents have a common understanding of the terminology and to be able to make concepts more similar for further analysis steps. For example one viewpoint agent may use the term 'ATM card' and another viewpoint agent just uses the term 'card' to refer to the same object (a state of correspondence), which is inserted into the ATM to get cash. One possibility is to show all terms, to decide which objects are synonyms, hyponyms and hypernyms and to replace them by one term. 'Replace' does not mean changing individual viewpoints. This can be done by introducing a table of synonyms, hyponyms and hypernyms. For example 'ATM card' and 'card' can be chosen from a list to see all sentences/objects containing these attributes. Thus one can decide if 'card' is always used as an 'ATM card'.

If synonyms, hyponyms and hypernyms are defined, the second step can be to determine if two concepts provide the same information. Thereby two or more concepts described by different viewpoint agents can be in consensus if they describe the same action or state using the same terminology. This is usually the case if viewpoint agents describe their viewpoints on the same level of detail. These concepts share the same node in a concept lattice and so are easy to identify.

Partial consensus is also possible. The concept 'A' can contain all attributes of the concept 'B' plus additional attributes. In a concept lattice this case is displayed in the way that concept 'B' is the subconcept of concept 'A'. For example the formal concept ({10-A, 13-B}{ATM, ejects, card}) is the superconcept of the concept ({13-B}{ATM, ejects, card, to customer}), where the first set is the objects identified by the sentence number and viewpoint and the second set are the attributes or parts of those sentences.

Partial consensus occurs if two or more formal concepts share some but not all attributes. For example the sentences {ATM, provide, receipt} and {ATM, release, receipt} share the attributes 'ATM' and 'receipt' and are thus subconcepts of the partial concept {ATM; receipt}. Thereby the same action or state can be described on a varying level of detail. For example the sentence {ATM, print, receipt, receipt show transaction number and transaction type and amount and account balance} gives more information than the sentence {ATM, releases, receipt} but they describe the same action.

The identification of different concepts using the same terminology will be difficult but the lattice may be able to assist. In such a situation some of the attributes/terms will be shared but others will not. This will suggest that the two viewpoints are referring to different things even if they use the same word/s. For example, the two sentences in Viewpoint A{customer, requests, receipt}{bank, issues receipt} and the sentence {customer, issues, receipt} in Viewpoint B use the same terminology but represent different concepts. There is obviously an error that needs to be reconciled by the viewpoint agents.

After the identification of concepts giving the same information the viewpoints can be investigated to find information not given in all viewpoints. Considering the flow of action these can be missing steps or a different sequence of action or states. Missing steps of the flow of action or missing conditions are represented by concepts which are not shared by all viewpoints. Information about a different sequence of actions or states can only be derived from the 'action/state#'. Where different levels of abstraction are used to specify requirements the analyst may choose to add or drop steps. However, the model that we are left with after negotiations is not expected to show all viewpoints now in total agreement. Many of the requirements will be shown as having come from a particular viewpoint but the group must agree to keep that requirement.

When many viewpoints are involved and/or the use case descriptions are long we have a number of strategies. We can do a pairwise comparison between two viewpoints and then add another viewpoint to the result until all viewpoints have been processed. Alternatively we consider all viewpoints concurrently but select a few sentences from each to compare at a time. Our tool allows selection of sentences to include based on one or more phrases or words. We also select sentences based on the content (done manually) to identify sentences covering similar steps in each use case. We have found these strategies to be a manageable way of handling scalability of our approach.

We provide a brief example of what a concept lattice looks like and how it can be read. Figure 7 shows a line diagram which includes the viewpoints of Agent A and Agent B for the "booking room" use case which is part of an online accommodation reservation system. Just the sentences concerning the system have been included. To read the line diagram start at the bottom nodes to find the agent who is the owner of the sentence, pick up the term in that node and then pick up all terms that can be reached by all ascending paths to get the complete sentence. For example the selected node on the far right represents the sentence that was written by agent A and says that the "system, shows, [the] room capacity". Most sentences are not shared by the two agents. However, we can see that Agent A and B agree that the System saves a request (fourth node from left). The sentences on the far left show that Agent A and B have a sentence stating that the system sends an email. In addition, Agent A states that the email is sent as a receipt. The diagram can be analysed for differences according to the 4-quadrant model by the project group to produce a more comprehensive and representative set of requirements that the participants will feel they have ownership of. By determining if the differences are due to terminology or the whether a difference concept is being considered we can apply a different strategy. For example, if the difference is due to terminology we use a synonym table. If the difference is due to a different concept being considered we may choose to delay or ignore the concept.

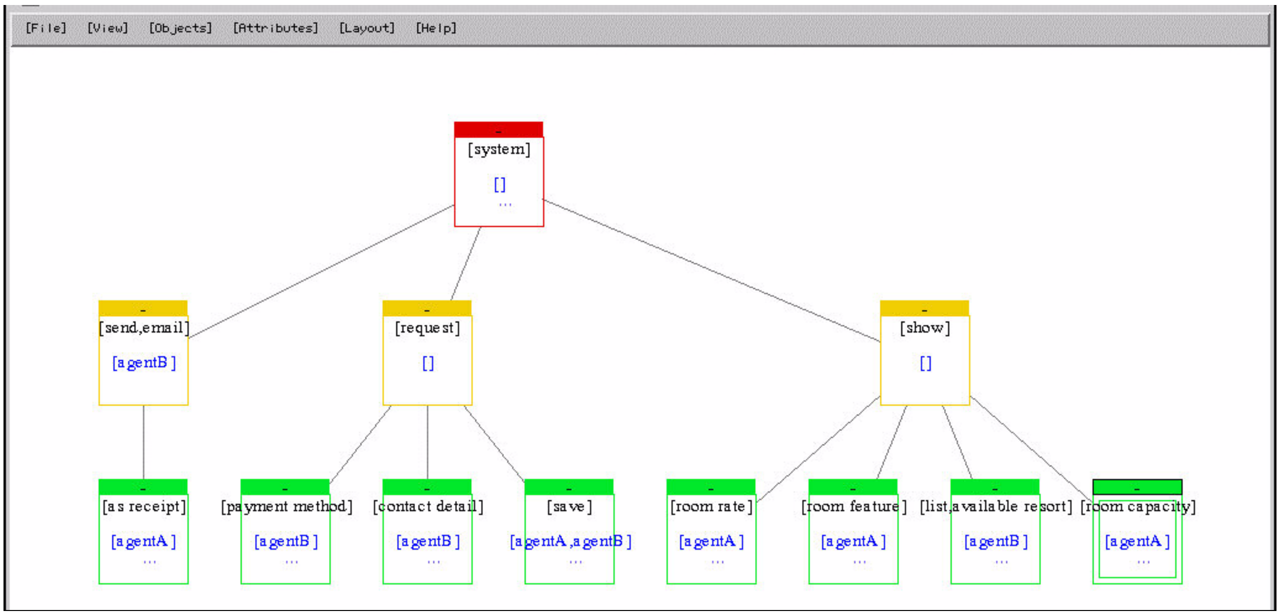


Figure 7: Line diagram for the "Booking Room" use case from Agent A and Agent's B viewpoints for sentences which concern the system.

3. Evaluation of the tool

In October 2002 we conducted a study to investigate the usefulness and feasibility of the reconciliation process in the RECOCASE tool and methodology. The study and our results are described next.

3.1 The Tests

We conducted three studies, each one involving three test participants who were all 3rd year computing students. For each study, the test participants attended three separate sessions. All the sessions were conducted during one day. The first session lasted for 1.5 hours, and the two following 45 minutes each.

During the two first tests the participants used the RECOCASE methodology and tool to create use cases for an automated video system (AVS) where the customer can self check out and rent videos from a video store. Test 3 was conducted with the same students as in test session 1, but this time they were given a new problem to solve. We organized the tests in this way to investigate any differences between first-time and more experienced users.

During the first session the participants were asked to fill out a pre-test questionnaire to collect descriptive data and identify any prior experience in using tools for use case writing. The motivation behind the RECOCASE approach and important concepts were explained, and the RECOCASE tool was demonstrated. The test participants then brainstormed together to discover use cases. One of us acted as group facilitator, led the sessions and drew the use case diagram on a board according to the test participants' directions.

After a one-hour break while the group facilitator entered data such as project name, users and use case names into the RECOCASE tool, the test participants had an individual session using the tool. The second session was a 45 minutes individual trial of the tool. Two use cases were chosen from the use case diagram, and the test participants were asked to write use case descriptions following our writing guidelines. The test participants had a one hour and 15 minutes break between the second and third session. During this break the group facilitator used the RECOCASE tool to create line diagrams where the nodes are selected words from the use case sentences. If some sentences have identical meaning but the use case authors have used different terms, the group facilitator inserted these words in the synonym table and later informed the group of the decisions she had made. Identifying synonyms reduces the number of nodes and thus produces a less complex and easier to read diagram. If there are many sentences, we developed several smaller diagrams to cover the sentences from each viewpoint.

The third session was scheduled for an hour, and the test participants used the diagrams to reconcile any differences and to produce a final set of use case descriptions which is representative of all the involved viewpoints.

We are currently planning to conduct another test where a new group of participants write a use case description without using the RECOCASE tool. We will then compare this use case description with the ones produced by test participants using the tool. By doing this we want to investigate whether participants using the tool produce longer and more detailed use case descriptions.

3.2 The Test Results

The rating of the usability of the RECOCASE tool was good (Figure 8). All usability attributes except efficiency and error recoverability were rated between 4 and 5 which means that they all ticked "agree" or "strongly agree" on the statements regarding the usability of the application. As can be seen in Figure 8, recovery from errors and perceived efficiency of the tool increased with each study.

We got feedback from the test participants in Test 1 saying that it was too hard and time consuming to enter the use case sentences. We modified the tool before Test 2 and the ratings were much better. Before Test 3 we also included some new icons to make it easier for the users to locate the command buttons. The participants in Test 3 had already tried the tool in Test 1 and wrote on the feedback forms that they liked the changes. They also rated the memorability of RECOCASE and all agreed or strongly agreed that is was easy to remember how to use the tool. We would like to improve the ratings on error recovery, and work is currently being done to make the application more robust.

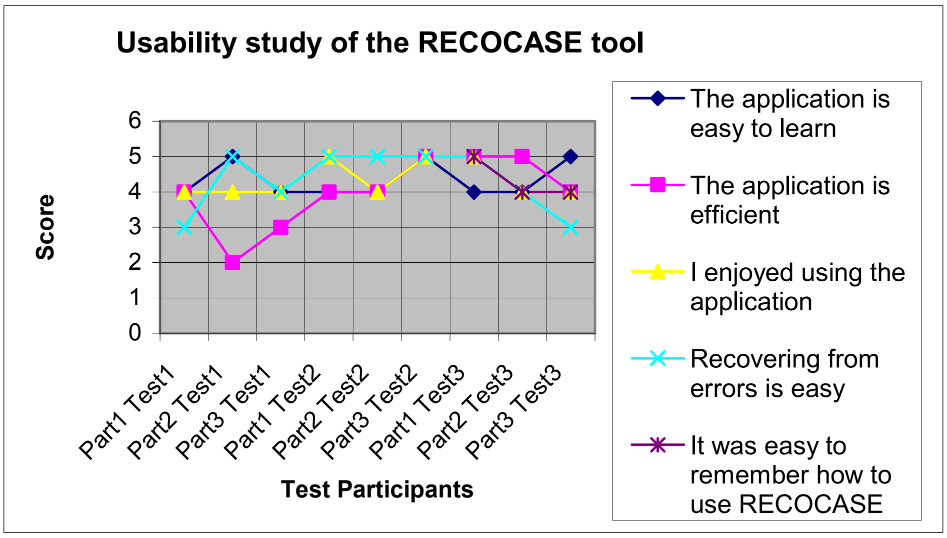


Figure 8: A line chart showing the participants rating of the RECOCASE tool's usability. 0 is "no opinion", 1 is " strongly disagree", 2 is "disagree", 3 is "undecided", 4 is "agree" and 5 is "strongly agree".

4. Conclusion

This paper has focused on presenting a tool that has been developed to implement the RECOCASE viewpoint development approach. We have also described how the guidelines and controlled language we have developed can be supported by the ExtrAns and RECOCASE-tool systems. We have described the design of our tool at the use case and class level and provided some of the system functionality. The tool also allows selection of sentences to include in a crosstable, mapping of terms, exploration of the crosstable and manipulation and navigation of the concept lattice.

The RECOCASE-tool is being extended with more features which shall constrain the viewpoint agent to compose viewpoints to following our guidelines. A restriction of the vocabulary for certain domains could be investigated whether it can improve the readability of concept lattices by the reduction of the number of concepts. Further exploration could also include the use of semantic roles and how they could be automatically assigned using an ontology. The work conducted to date provides a solid foundation for these future enhancements and endeavours.

Acknowledgements: Many thanks to Oscar Aguilera for his hard work on the RECOCASE-tool. Thanks to Rolf Schwitter and Diego Molla-Alloid for their assistance with ExtrAns and other language aspects of this project. Thanks for Rolf for thinking up the RECOCASE name.

Bibliography

Boettger, K. (2001) Modelling and Reconciling Functional Requirements from Different Viewpoints using Use Cases/ Scenarios and Formal Concept Analysis *Masters Thesis,* University of Mannheim.

Boettger, K., Schwitter, R., Richards, D., Aguilera, O. and Molla, D. (2001) Reconciling Use Cases via Controlled Language and Graphical Models, The Proceedings of the 14th International Conference on Applications of Prolog, (INAP'2001), 20-22 October, 2001, University of Tokyo, Japan, 1860195.

Constantine, L. L. and Lockwood, L. A. D., (1999) *Software for Use : A Practical Guide to the Models and Methods of Usage-Centered Design*, ACM Press.

Cox, K., (2001) Experimental Material, http://dec/bournemouth.ac.uk/staff/kcox/UCwriting.htm

Darke, P. and Shanks, G., (1997) Managing User Viewpoints in Requirements Definition, *8th Australasian Conference on Information Systems*.

Easterbrook, S. and Nuseibeh, B. (1996) Using Viewpoints for Inconsistency Management BCSEEE Software Engineering Journal January 1996, 31-43.

Finkelstein, A.C.W., Goedicke, M., Kramer, J. and Niskier, C. (1989) Viewpoint Oriented Software Development: Methods and Viewpoints in Requirements Engineering In *Proceedings of the Second Meteor Workshop on Methods for Formal Specification* Springer Verlag, LNCS.

Fromkin, V., Rodman R., Collins, P. and Blair, D. (1996) An introduction to language, 3rd edition, Harcourt Brace & Company, Australia.

Jacobson, I. (1992) *Object-Oriented Software Engineering*, Addison-Wesley.

Mullery, G. P. (1979) CORE - a method for controlled requirements expression *In Proceedings of the 4th International Conference on Software Engineering (ICSE-4)*, IEEE Computer Society Press, 126-135.

Richards, D. and Menzies, T. (1998) Extending the SISYPHUS III Experiment from a Knowledge Engineering to a Requirement Engineering Task, *11th Workshow on Knowledge Acquisition, Modeling and Management*, SRDG Publications, Departments of Computer Science, University of Calgary, Calgary, Banff, Canada, 18th-23rd April, 1998

Richards, D. and Zowghi, D. (1999) Maintaining and Comparing Requirements, *Proceedings of the Fourth Australian Conference on Requirement Engineering ACRE'99*, Macquarie University, Sydney, 29-30 September, 1999

Rolland, C. and Achour, C. B. (1998) Guiding the Construction of Textual Use Case Specifications", *Data & Knowledge Engineering Journal*, Vol 25, No 1-2, pp. 125-160, North Holland, Elsevier Science Publishers. March 1998.

Schwitter, R. and Fuchs, N. (1996) *Attempto From Specifications in Controlled Natural Language towards Executable Specifications*, Institut fuer Informatik der Universitaet Zuerich, 1996

Shaw, M. L. G. and Gaines, B. R. (1988) A Methodology for Recognising Consensus, Correspondence, Conflict and Contrast in Knowledge Acquisition System, *Proceedings of the 3rd Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada, 1988.

Sleator, D. D. and Temperley, D. (1991) Parsing English with a Link Grammar, *Technical Report CMU-CS-91-196*, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, 1991

Wille, R. (1982) Restructing Lattice Theory: An Approach Based on Hierarchies of Concepts, *Ordered Sets*, D. Reichel, Dordrecht, pp. 445-470.

Wille, R. (1992) Concept Lattices and Conceptual Knowledge", *Computers and Mathematics with Applications*, 23, pp. 493-522.

Wirfs-Brock, R. (1993) Designing Scenarios: Making the Case for a Use Case Framework. *Smalltalk Report* Nov-Dec 1993. NY, NY: SIGNS Publications.